Doctoral theses at NTNU, 2022:61

Sivert Bakken

Development of a Small Satellite with a Hyperspectral Imaging Payload and Onboard Processing for Ocean Color

Doctoral thesis

NTNU
Norwegian University of Science and Technology
Thesis for the Degree of Philosophiae Doctor
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of Science and Technology

Sivert Bakken

Development of a Small Satellite with a Hyperspectral Imaging Payload and Onboard Processing for Ocean Color

Thesis for the Degree of Philosophiae Doctor

Trondheim, March 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of Science and Technology

Thesis for the Degree of Philosophiae Doctor

Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

ISBN 978-82-326-5963-0 (printed ver.)
ISBN 978-82-326-5356-0 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

Doctoral theses at NTNU, 2022:61

Printed by NTNU Grafisk senter

Abstract

This thesis presents a CubeSat system that can capture ocean color by hyperspectral remote sensing, and how this can be exploited as a surveying technology that can aid in ocean observation and other related objectives. The system uses a hyperspectral camera, i.e., a camera that can capture an image across the full spectrum from visible to near-infrared light. The thesis focuses on the concept of the system, how the different data products could be processed to observe the ocean efficiently, and how the system and payload can be designed to accommodate changing requirements. These objectives raised the following research questions.

- **RQ-1.** How can a CubeSat platform be developed for hyperspectral ocean color applications and provide end-users with valuable information?
- **RQ-2.** What kind of algorithms or models for better hyperspectral data acquisition and reduced data latency can be deployed on-board a CubeSat tailored for ocean color observations?
- **RQ-3.** How can a CubeSat be used to support the development, validation, and upgradeability of new in-orbit data processing algorithms?

For **RQ-1** a CubeSat concept is presented. This thesis only provides details on the potential on-board processing schemes and references the CubeSat concept's other aspects. The appendix provides further details of the CubeSat specifications.

For **RQ-2** some different processing strategies are presented. This thesis attempts to answer the research question in a general manner only concerning hyperspectral image processing and some considerations regarding simplistic models explicitly developed for ocean color applications and parameter retrieval.

For **RQ-3** the latter part of the thesis presents how the related challenges were reduced to create a CubeSat with the desired capabilities at an acceptable risk.

This thesis is not an extensive exploration of all there is to know about these complex topics. Instead, it is an overview of the considerations taken when developing HYPSO-1 and how this CubeSat concept can be used with other sensor platforms.

All in all, this work documents the creation of a CubeSat that could and should give new insights into how to do operational ocean color observations in the future.

Sammendrag

Denne oppgaven presenterer et CubeSat-system som kan fange havfarger ved hyperspektral fjernmåling, og hvordan dette kan utnyttes som en kartleggingsteknologi som kan hjelpe til med havobservasjon og andre relaterte mål. Systemet bruker et hyperspektralt kamera, det vil si et kamera som kan ta et bilde over hele spekteret av synlig og nær-infrarødt lys. Oppgaven fokuserer på konseptet til systemet, hvordan de ulike dataproduktene kan behandles for å observere havet effektivt, samt hvordan systemet og nyttelasten kan designes for å møte endrede krav. Disse målene reiste følgende forskningsspørsmål.

- **RQ-1.** Hvordan kan en CubeSat-plattform utvikles for hyperspektrale havfargeapplikasjoner og gi sluttbrukere verdifull informasjon?
- **RQ-2.** Hva slags algoritmer eller modeller for bedre hyperspektral datainnsamling og redusert dataforsinkelse kan brukes i en CubeSat skreddersydd for havfargeobservasjoner?
- **RQ-3.** Hvordan kan en CubeSat brukes til å støtte utvikling, validering og oppgraderingsevne til nye databehandlingsalgoritmer i bane?

For **RQ-1** presenteres et CubeSat-konsept. Denne oppgaven gir kun detaljer om potensielle prosesserings strategier som kan brukes i bane og kun refererer til de andre aspektene ved CubeSat-konseptet. Vedlegget gir ytterligere detaljer om CubeSat-spesifikasjonene.

For **RQ-2** presenteres noen forskjellige prosesserings strategier. Denne oppgaven forsøker å besvare forskningsspørsmålet på en generell måte som kun tar for seg hyperspektral bildebehandling og gir i tillegg noen betraktninger angående forenklede modeller som er eksplisitt utviklet for havfargeapplikasjoner og parameterinnhenting.

For **RQ-3** presenterer siste del av oppgaven hvordan noen av utfordringene ble mitigert for å lage en CubeSat med de ønskede egenskapene på en måte som innehar et akseptabelt nivå av risiko.

Denne oppgaven er ikke en omfattende utforskning av alt det er å vite om disse komplekse temaene. I stedet er det en oversikt over hensynene som er blitt tatt ved utvikling av HYPSO-1 og hvordan dette CubeSat-konseptet kan brukes med andre sensorplattformer.

Alt i alt dokumenterer dette arbeidet skapelsen av en CubeSat som kan og bør gi ny innsikt i hvordan man kan gjøre operasjonelle havfargeobservasjoner i fremtiden.

Preface

An underlying desire to understand and learn more about the world we live in has given me the drive to pursue this Ph.D. When I first applied for the opportunity as a 22-year old fourth-year student in the integrated master program for engineering cybernetics, I did not know what I was getting into. Since then, I have learned a lot about our world and myself.

Undergoing a Ph.D. can be challenging for many reasons, and the emergence of a global pandemic did not help. The impacts of this pandemic unquestionably halted some of the planned research activities.

The original title of the Ph.D. project that I applied for was “Coordinated oceanographic observation system with autonomous aerial/surface robots and hyper-spectral imaging in SmallSat”. The project was ambitious and concerned a complex system of systems. Most of the platforms considered in that title were not at a technical readiness level where they could be coordinated as a system of systems when I started. For example, the satellite was still in the concept phase.

These limitations meant that I had to learn a lot about a lot that was not covered as part of my educational background at the point of starting. Making a satellite, even just parts of the payload software envisioned, required many resources. My efforts have been focused on realizing the actual satellite and how it could be used. It remains as future work to perform more simulations and experiments of the system of systems. I am very grateful that I was allowed to pursue this Ph.D. focusing on the hyperspectral payload and its software.

Acknowledgments

This thesis is a product of nearly three and a half years of hard work, and many people deserve to be mentioned because of their involvement. I want to use this opportunity to thank my supervisors, friends, colleagues, and family for their support during this time in my life.

Thank you, Tor Arne Johansen, for motivating me, guiding me when needed, and providing insight and ideas for different research areas. Similarly, Geir Johnsen, thank you for always sharing your enthusiasm and engagement within the field of ocean biology, bio-optics, and remote sensing. I am grateful for having you two as supervisors, and I have learned a lot from you both professionally and personally. Thank you, Harald Martens, for being my mentor during the Ph.D., it has been amusing and a good learning experience to discuss different topics related to hyperspectral imaging with you. And I could not end this paragraph without mentioning Milica Orlandic, my unofficial supervisor, since I did my master's. I will be forever grateful for your ability to see both people and the academic work being done.

I am happy to have been a part of the NTNU SmallSat Lab during its startup. This lab has been an excellent place for me to do more than just research; it has provided me with an arena to establish new friendships, learn a lot about satellites, play around with 3D printing and learn so much from so many different and unique people.

I am grateful for all the discussions we have had together, the things I learned, being part of a team, and all the fun moments that accompanied this time in my life. Thank you Evelyn Honoré-Livermore, Joseph L. Garrett, Elizabeth F. Prentice, Roger Birkeland, Dennis D. Langer, Marie Bøe Henriksen, Gara Q. Diaz, Mariusz E. Grøtte, Bjørn A. Kristiansen and Amund Gjersvik. Special thanks to Evelyn Honoré-Livermore for being an incredible project leader and guiding me through my responsibilities regarding the software development of the payload for our initial satellite. Similarly, I would like to thank Joseph L. Garrett, Roger Birkeland, and Dennis D. Langer for their crucial contributions to the software development and good discussions concerning making the satellite more operational. Lastly, I must mention Elizabeth F. Prentice; thank you for all the small talks we had in the lab while you were getting the actual payload ready.

Thank you to all the students who have been a part of our satellite team. Without you, we would not even be close to having a satellite! Special thanks to Live Jacobsen for making the Front-page illustration.

To my friend and mentor Ajit Subramaniam, thank you for all the helpful discussions we have had about the research cruise you let me join and all the ideas you have shared with me on how we might use the satellite we have made.

To Kristin Johansen, I will never be able to say how truly appreciative I am that you have taken the time to read this thesis and helped me present it in a way that is more accessible. Without your thorough eye, this piece of work would have been lesser.

I would also like to acknowledge all the support I have received from the department. To Tove Kristin Blomset Johnsen, thank you for all the help you provided when I struggled the most with my Ph.D. To all my colleagues and lunch-friends at the department, Sverre Velten Rothmund, Pål Holthe Mathisen, Marianna Wrzos-Kaminska, Jostein Løwer, Stefano Brevik Bertelli, Johann Alexander Dirdal, Kristoffer Gryte, Martin Lysvand Sollie Eirik Lothe Foseid, and everyone else.

Thank you to my friends outside academia for providing me with a free space to escape to think about other things. I would like to direct a special thanks to Videokomiteen at Samfundet for the social group that this volunteer work has given me through my years at Norwegian University of Science and Technology (NTNU). Furthermore, I would like to thank my friends from Hønefoss who live in Trondheim, Erlend Ravlo, Eirik Sande, Torkel Forbord, and Baldur Kjelsvik, as well as Nils Barlaug (even though you are from Oslo) for our social gatherings and all the fun times we have had. I would also like to direct a special thanks to Adrian Austevoll; I am so grateful that you are always just a phone call away even though England, after you moved there during the pandemic, seems quite far. I will always treasure our conversations, and how you help me tone down my perception of my own problems by sharing some of yours.

To my dear family, my mother and father, Anne and Arve, thank you for always supporting me in the ambitious challenges that I seek. It has not always been easy, and none of us has understood or been familiar with what I have embarked upon. However, it seems like I am coming out on the other end of this adventure, and that would not have been remotely possible without your love and care. Thank you so much.

Finally, dear Aurora, thank you for being the supportive and generous person you are. You always guide me to be better, and you are always there when I need you. You are my muse and my inspiration. Thank you.

Funding Information

The work that constitutes this thesis was carried out at the Department of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU) in Trondheim, Norway, from August 2018 to February 2022. These studies have been performed under the supervision of Professor Tor Arne Johansen (NTNU), and co-supervisor Professor Geir Johnsen (NTNU). The Norwegian Research Council supported this work through the Centre of Autonomous Marine Operations and Systems (NTNU AMOS) (grant no. 223254), the MASSIVE project (grant no. 270959), and the European Space Agency (PRODEX - 4000132515). NO Grants 2014 – 2021, under Project ELO-Hyp, contract no. 24/2020.

Supervision:

During the doctoral study, I have been co-supervisor or strongly involved in the supervision of 9 master students; Monica Lapadatu, Đorđije Bošković, Magnus Danielsen, Ole Martin Borge, Esmée Oudijk, Tuva Moxnes, Torbjørn Bratvold, Simen Netteland, Linn Marie Sønstetrud, Thomas Halvard Bolle and Kristine Døsvik. Furthermore, I have had the privilege of supervising and leading the software development of the HYPSO-1 Hyperspectral Payload, where more than 20 more students have been involved during this period.

I have also participated in the EN640 research cruise. During this research cruise I did investigative work under the supervision of Professor Ajit Subramaniam. The results are still a work in progress, and are presented in chapter 4.

Acronyms

[Chl a] Chlorophyll-a concentration ($\mathrm{mg}/\mathrm{m}^{-3}$).

AC Atmospheric Correction.

AMOS Centre for Autonomous Marine Operations and Systems.

API Application Programing Interface.

AUV Autonomous Underwater Vehicle.

BOUSSOLE buoy for the acquisition of a long-term optical time series.

CAN Controller Area Network.

CDOM Colored Dissolved Organic Matter.

CI Continuous Integration.

CLI Command Line Interface.

COTS Commercial-Off-The-Shelf.

CPS Cyber-Physical Systems.

CPU Central Processing Unit.

CSP Cubesat Space Protocol.

CTD Conductivity for salinity, Temperature, and Depth for pressure of seawater.

CZCS Coastal Zone Color Scanner.

DE Digital Engineering.

ECSS European Cooperation for Space Standardization.

EM Engineering Model.

EPS Electrical Power Unit.

ESA European Space Agency.

FC Flight Computer.

FM Flight Model.

FOV Field-Of-View.

FPGA Field Programmable Gate Array.

FT File Transfer.

GIOP Generalized Inherent Optical Properties.

GPU Graphics Processing Unit.

GS Ground Station.

GSE Ground Support Equipment.

HAB Harmful Algal Blooms.

HICO Hyperspectral Imager for the Coastal Ocean.

HIL Hardware-In-the-Loop.

HIPP Hyperspectral Image Processing Pipeline.

HPLC High-Performance Liquid Chromatography.

HSI Hyper Spectral (Imager/Image).

HYPSO HYPer-spectral SmallSat for Ocean observation.

ICD Interface Control Document.

INCOSE International Council On Systems Engineering.

IOP Inherent Optical Properties.

LBA Light Beam Attenuation.

Least Absolute Shrinkage and Selection Operator LASSO.

MASSIVE Program for Mission-oriented Autonomous Systems with Small Satellites for Maritime Sensing, Surveillance and Communication.

MBSE Model-Based Systems Engineering.

MCE Model-Centric Engineering.

MCS Mission Control System.

MDR Mission Design Review.

MERIS MEdium Resolution Imaging Spectrometer.

MIL Model In the Loop.

ML Machine Learning.

MOBY Marine Optical BuoY.

MVP Minimum Viable Product.

NASA National Aeronautics and Space Administration.

NIFRO Norwegian Industrial Forum for Space Activities.

NN Neural Network.

NNG Nanomsg Next Generation.

NTNU Norwegian University of Science and Technology.

OBIP On-board Image Processing.

OPU On-board Processing Unit.

OS Operating System.

PC Payload Controller.

PCA Principal Component Analysis.

PLS Partial Least Squares.

PLSR Partial Least Squares Regression.

PM Project Management.

PR Pull Request.

PSU Power Supply Unit.

QAA Quasi-Analytical Algorithm.

QM Qualification Model.

RF Radio Frequency.

RGB Red-Green-Blue Color Channels.

RT Radiative Transfer.

SAA Semi-Analytical Algorithm.

SBA Skylight-Blocked Approach.

SDR Software Defined Radio.

SE Systems Engineering.

SEE Single Event Effect.

SERC Space Engineering Research Center.

SIL Software In the Loop.

SNR Signal-to-Noise Ratio.

SOA Service-Oriented Architecture.

SoC System-on-Chip.

SoS System of Systems.

SP SpectroPhotometer.

ToA top of atmosphere.

TSM Total Suspended Matter.

UAV Unmanned Aerial Vehicle.

UI User Intrerface.

USART Universal Synchronous and Asynchronous Receiver-Transmitter.

USV Unmanned Surface Vehicle.

XP Extreme Programming.

Contents

Chapter 1

Introduction

> *We need to respect the oceans and take care of them as if our lives depended on them. Because they do.*
>
> Sylvia Earle

Our world is changing, and the relevance of resources and services related to aquaculture and global aquatic environments is increasing due to population growth and climate change [1, 2]. New technologies for surveying the ocean and other water bodies can provide science-based information to support management and policy strategies. The use of new, improved, and augmenting surveying strategies is a step in reaching the sustainability goals related to water set out by the United Nations [3]:

- 6: Ensure availability and sustainable management of water and sanitation for all.
- 14: Conserve and sustainably use the oceans, seas, and marine resources for sustainable development.

This thesis presents a CubeSat system that can capture ocean color by remote sensing and how this can be exploited as a surveying technology that can aid in reaching the sustainability goals and other objectives. The system uses a hyperspectral camera, i.e., a camera that can capture an image across the entire spectrum from visible to near-infrared light.

This introductory chapter presents the topics of this thesis such that their common theme becomes clearer. The topics range from hyperspectral image processing, how these can be tailored towards ocean monitoring and sampling, and to accommodate these through the use and development of said CubeSat system. This chapter emphasizes ocean monitoring in Norway, the Norwegian coast, and the Norwegian industry and its potential impacts, while the following chapters are more general.

The Program for Mission-oriented Autonomous Systems with Small Satellites for Maritime Sensing, Surveillance and Communication (MASSIVE) project combined with the HYPer-spectral SmallSat

for Ocean observation (HYPSO) project is vital components in a novel disruptive approach for more effective marine ecosystem research and monitoring. By combining data from existing platforms focusing on low-altitude and in-situ observations from buoys, ships, and other autonomous vehicles with hyperspectral imaging from small satellites, the goal is to achieve new insights at unprecedented temporal and spatial scales.

This project is made possible by NTNU's long-term commitment to research and existing infrastructure within its Center of Excellence on Centre for Autonomous Marine Operations and Systems (AMOS). This research infrastructure provides significant synergies that bring together leading scientists in remote sensing, autonomous systems, hyperspectral images and imagers, small satellite systems, ocean modeling, bio-optics, biogeochemistry, and ecology.

1.1 Hyperspectral Remote Sensing

A normal Red-Green-Blue Color Channels (RGB)-camera collects three wavelength bands in the visible range of the electromagnetic spectrum (400-700 nm), i.e., a three-channel image. The term hyper in hyperspectral refers to the spectral resolution of the instrument or sensor in use. There is no strict definition of when a sensor becomes hyperspectral, but it is often characterized by a high spectral resolution, such as < 10 nanometer [4]. A hyperspectral camera can sample signals from the electromagnetic spectrum at many distinct wavelengths. This is usually in the visible and the near-infrared range (400-2500 nm), as opposed to RGB or multispectral, which sample fewer bands and often with a broader bandwidth, i.e., more parts of the spectrum is associated with a pixel.

Figure 1.1 gives an illustration of the working principle of the Hyper Spectral (Imager/Image) (HSI) acquisition technique known as push-broom scanning. The technique works by dispersing the Collimated light that passes through a thin slit across an imaging sensor through various optical components. More details on the HSI payload found in HYPSO-1 is given in [5]. Through moving the sensor platform, the imager can scan a given area and collect both spatial and spectral information. This results in an image cube that can be used for further analysis.

Remote sensing refers to measuring the properties of an object or material without being in contact with it [4]. These properties can range from biological, geophysical, or chemical. Through analysis of the radiative measurements, e.g., measurements of light, it is possible to infer Inherent Optical Propertiess (IOPs), the properties that are unique to that object or material in its current state [7]. Remote sensing is often verified with in-situ measurements when the instrumentation needs to be in contact with the object or material of interest to collect measurements. Thus, remote sensing is particularly suited for data collection over large areas at a high temporal resolution. When the object or material of interest is out of reach, it is hard to get close to, or the process itself would be affected by physical contact.

Generally speaking, for hyperspectral images, the number of spectral bands will be much higher than the number of parameters that can be retrieved. This property is due to the high correlation between the measurement next to each other in the electromagnetic spectrum [1, 4]. However, with

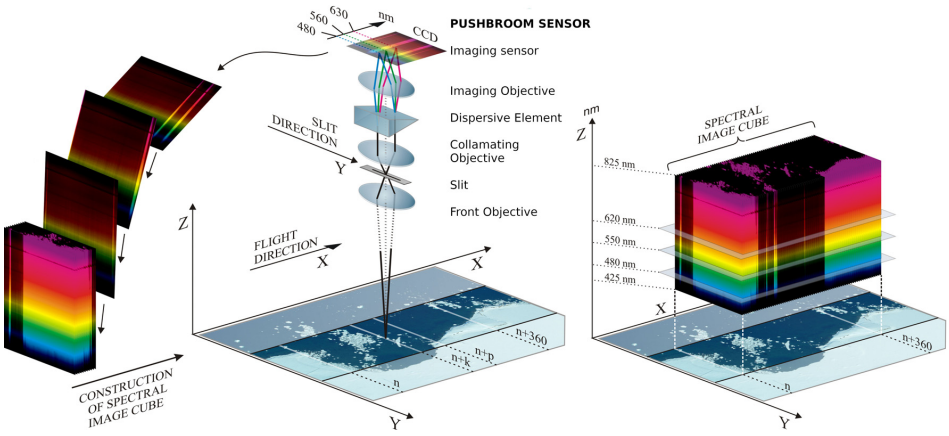


Figure 1.1.: Illustration of the working principles of an HSI using the push-broom technique. For details see [5]. Original illustration by Dr. Zsolt Volent, SINTEF Ocean, adapted from [6] with permission.

more hyperspectral data across a wide variety of conditions, specifically for inland and coastal water applications, we may get new information in parts of the spectrum that have previously been given less attention. Furthermore, hyperspectral data is expected to limit the uncertainty in the retrieved parameters when compared to using fewer spectral bands [1, 8].

This thesis investigates only a subset of the approaches that can be used to gain more information throughput from hyperspectral data. A more detailed overview of the potential avenues to explore and exploit hyperspectral data for aquatic remote sensing is given in [1]. Using the algorithm naming scheme found in [1], the chapters 5 and 6 attempts to best utilize coordinate transformations of the spectra, while chapters 7 and 8 investigates how spectra can be used as predictors through the use of neural networks and parametric regression [9]. In chapter 5 the results from the coordinate transform are also explored by using target detection as a potential use case after the coordinate transform. In chapter 6 the limitations of coordinate transforms for compression is compensated for by analyzing the residuals.

1.2 Monitoring of Ocean Color

The ocean plays a crucial role in our world's climate and ecosystems, yet it is one of the least explored environments on Earth. It is a complex and rapidly changing subject to investigate but can be a hazardous environment. Despite these challenges, there is a growing need to understand how these systems interact. The data collected from ocean monitoring should be used to understand our impact on the world's oceans, and our future [2]. In [10] it is shown how extensive data records of various types of data regarding water bodies can be used to uncover limitations of our knowledge on the ocean systems that surround us. This again shows the importance of creating, ordering, and keeping such records.

For centuries, understanding the ocean has been a topic for oceanographers, and different novel techniques to say something about its state have been developed. It can, however, be challenging to sample systematically and across large areas in the ocean as it is very dynamic. In the past decades, the instruments used to understand the ocean have become more sophisticated. Integrating measurements from many different sources has helped elevate our understanding of the complex systems at play and improve our ocean knowledge. These ocean systems are, among other things, responsible for most of the oxygen production on Earth and the carbon dioxide uptake. It is vital not to impede this process further. The challenge of systematically sampling large areas can be resolved by combining measurements from ocean color satellites capturing the visual to near-infrared range of the electromagnetic spectrum, with measurements from other observation platforms and coastal observatory networks [7, 11].

Norway is an ocean nation, where fisheries, aquaculture, oil and gas, and other activities along the coast have a significant role in our economy, society, and politics. Along the coast of Norway lies the Norwegian continental shelf. The area is depicted in Figure 1.2. Norway has sovereign rights of this continental shelf, which is an area four times that of the Norwegian mainland. The area is rich in natural resources such as petroleum and gas, important for primary and secondary production in Estuarine resource, e.g., fish, shellfish, mollusks, crabs, seagrass beds, oyster reefs, and many other organisms, and is thus essential for aquaculture. The aquaculture industry is expected to have increased importance in the years to come. There is much activity along the Norwegian coast, and thus the Norwegian continental shelf is of great economic interest to Norway.

1.2.1 Former Ocean Monitoring Schemes in Norway

Before satellite systems for ocean color monitoring became operational, phytoplankton blooms and other critical environmental variables were monitored by other means. One of the methods used is water samples taken in areas of interest, for example, by research cruises going out to sea or samples at strategic areas along a coast [11]. The benefit of these direct in-situ samples is that it can be less uncertainty surrounding the quality of the measurements, as there are fewer potential sources of disturbance.

Collecting and processing water samples using, e.g., High-Performance Liquid Chromatography (HPLC) is not suitable for sampling over large areas nor at high frequencies, as this can be both time-consuming and expensive. However, it is still deemed a vital activity for validation of operational remote sensing algorithms [11]. This is why HPLC is still the standard method used to provide ground truth for important biomarkers, such as Chlorophyll-a concentration ($\mathrm{mg}/\mathrm{m}^{-3}$) ([Chl a]) when developing algorithms for remote sensing. Quality ground truth data is essential to be able to develop better.

A combination of ground and aerial-based sampling strategies is used today to get the desired insight about the ocean state. However, combining these measurements to gain valuable information requires significant planning and resources.

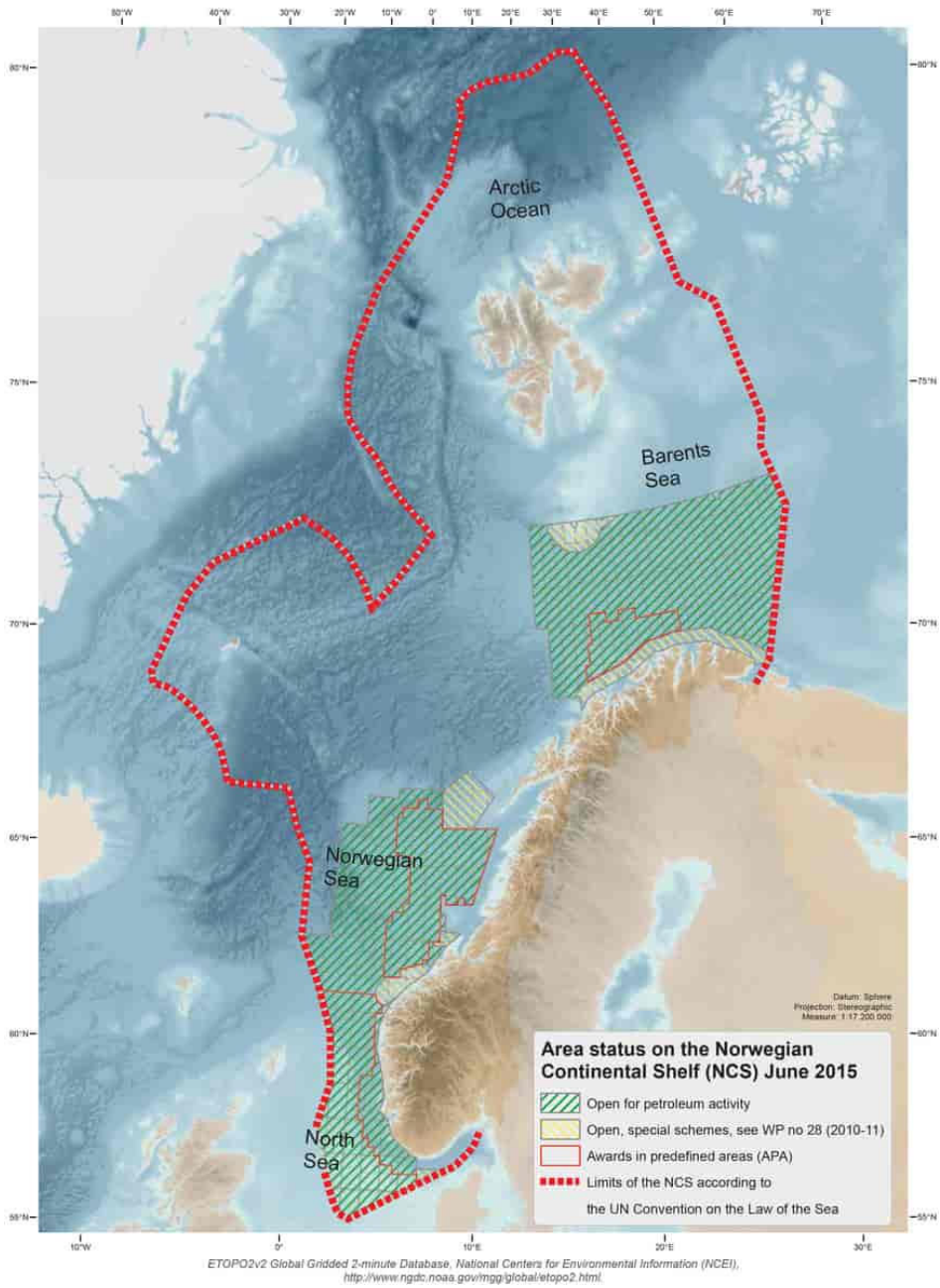


Figure 1.2.: The Norwegian continental shelf, image courtesy of Oljedirektoratet [12]

The SeaWatch buoys and Observer Network

The former advanced buoy system along the Norwegian coast is an example of a system that made many measurements and understood them in a unified context. This system was developed by OCEANOR and was known as SeaWatch. The SeaWatch system was operational from 1988 to 1993 [11, 13].

This Seawatch system was made up of multiple buoys placed in areas of particular interest along the Norwegian coast. The system provided in-situ monitoring and forecasting of physical, chemical, and biological variables in near real-time while it was operational. The provided data consisted of time series of Light Beam Attenuation (LBA) measurements for phytoplankton and zooplankton biomass, oxygen saturation, temperature, salinity, as well as the speed and direction of winds and currents, along the Norwegian coast.

In addition, the system was used as part of an observer network, consisting of the local fish farms and the Food Hygiene Control Authorities. This extended observer network provided data on parameters such as turbidity and watercolor in terms of Secchi-disc depth, sea surface temperature, general weather conditions. This observer network reported on fish behavior and feeding, as well as any unusual biological events. Water samples were routinely sent to the relevant authorities for analysis to identify and enumerate phytoplankton from these nodes in the observer network. This activity had a special focus on Harmful Algal Blooms (HAB) species.

This system did unquestionably provide information on the state of the ocean at areas of great interest. However, it may be challenging to justify the operational costs of such a system, given that it was discontinued. It was discontinued due to few HABs in Norway in the previous decade (2010-2020).

Ships of Opportunity and the Ferrybox

In [11] a case example of how coastal environmental monitoring and operative phytoplankton monitoring can be achieved with existing instrument platforms is given. According to [11], integration of data from different platforms that are complementary in terms of spatial and temporal coverage is necessary to establish an operative environmental monitoring system that can be used in coastal areas. That is, measurements from ocean color satellites will need to be combined with other manual, sensor carrying buoys, or ship-based measurements to cover the larger spatial areas with a sufficient revisit time and the desired precision and accuracy of the measurements. In [11] three methods of collecting data are identified to make the proposed monitoring system operational.

Firstly, there is still a need to collect coastal water samples manually and regularly, send them to a central facility for manual identification of algal groups and species, and use this expertise to determine if they are benign or potentially harmful. This first method will also validate other algorithms used on the data collected by other means.

The Ferrybox is a continuous flow-through system or concept to identify and quantify objects of interests in water, that is designed to be installed on ships in coastal traffic and can collect turbidity and [Chl a] measurements. This Ferrybox system can be installed on *any* ship that is in regular traffic across an area of interest. However, the travel route of such *ships-of–opportunity* will not choose its path based on the data it is collecting.

The last set of measurements integrated with the Ferrybox and manual water samples, [11] comes from the ocean color satellite images from MEdium Resolution Imaging Spectrometer (MERIS). The MERIS data was then rendered as RGB images and the turbidity and [Chl a] products were seen in combination with the other measurements sampled during the described project.

Suppose any new or alarming environmental conditions are observed through manual water samples, the Ferrybox system, or ocean color satellite images. In that case, the collected data should be processed with algorithms tailored to local conditions [7, 11]. By using ocean color satellite images from MERIS, a multispectral instrument on-board European Space Agency (ESA)'s Envisat platform, [11] demonstrates how to identify the spatial extent of a bloom, as well as the bloom patchiness, and the transport of a bloom. Furthermore, [11] emphasizes that these ocean color satellite images are valuable when doing large-scale observations.

A potential future ocean monitoring schemes to be used in Norway is briefly presented in section 1.3, and the HYPSO-1 satellite, representing the apex of the observational pyramid given in Figure 1.4, is detailed in chapter 2.

1.2.2 Remote Sensing of Ocean Color

Many oceanographic parameters can be inferred from radiance measured by satellite-based sensors and ocean color specifically. The sunlight measured at the top of the atmosphere by the satellite sensors has passed through the atmosphere, been reflected, absorbed, and scattered by different constituents found in the ocean, and then transmitted back through the atmosphere. In other words, there are a lot of various factors affecting the signal and noise characteristics, and some of these are briefly discussed in chapter 7

Compensation of the effects on optical signals from the atmosphere, adjacency effects from neighboring landmasses, and influences from the ocean surface is an active area of research [4, 7, 14, 15]. However, after appropriate compensation of known effects, the resulting water-leaving radiance can, in most cases, be used to determine the spectral scattering and absorption properties of the remotely sensed dissolved and suspended materials and Colored Dissolved Organic Matter (CDOM). Some current limitations of one popular model for IOPs retrieval [7], used after appropriate compensation of known effects, is briefly discussed in chapter 4.

The captured spectral information is dependent on the alterations of the radiance made by the oceanic constituents [7]. These alterations have traditionally been used to determine such parameters as [Chl a] directly through empirical relationships that relate to the desired parameter [7, 16]. Furthermore,

as ocean color is determined by IOPs, the captured spectral information can also be used to derive these IOPs, and that, in turn, can be used to estimate chlorophyll concentration, as well as a myriad of other oceanic parameters of interest [7].

Ocean Color Satellites

Ocean color remote sensing by spacecraft has been carried out since the launch of the Coastal Zone Color Scanner (CZCS) in 1978, and the many satellites following it [17]. These initial satellites showcased that the ocean color observed from space was suited to infer [Chl a] at a much larger, even global scale than previously available with sufficient accuracy and precision [2]. Some prominent satellites and their properties are given in Figure 1.3, with HYPSO-1 included using the original flight badge.

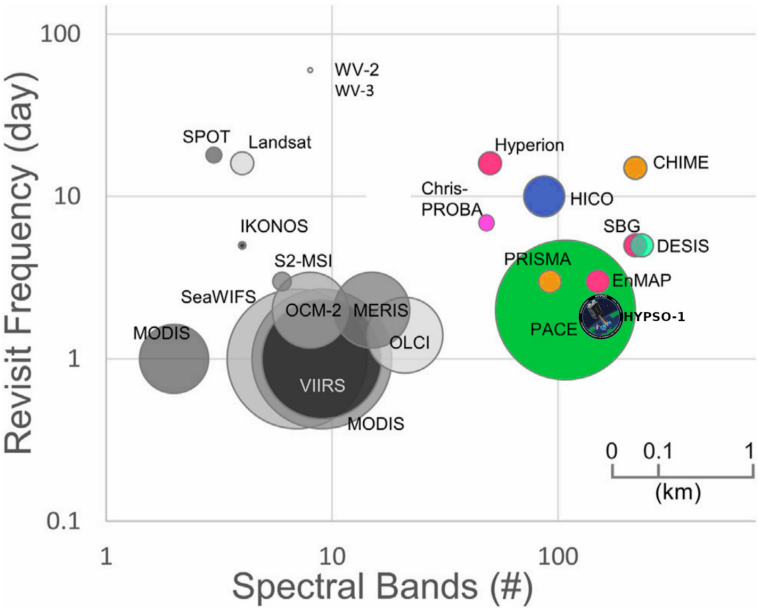


Figure 1.3.: A plot of several missions depicting the trade-off between the number of spectral bands, revisit frequency along the axis, and the approximated spatial footprint as the size of the circle. Here multispectral missions are presented in gray, while hyperspectral missions are presented with colors. This illustration is adapted, with permission, from [1]. The HYPSO-1 mission is marked by the HYPSO logo.

The data provided by ocean color satellites have advanced certain scientific fields such as biological oceanography, biogeochemistry focusing on ocean composition, physical oceanography, and ocean-system modeling. Additionally, the ocean color data has helped manage fisheries and coastal management.

Data from ocean color satellites is often used to improve models for dynamic ocean modeling and models of different ecosystems [2, 11], combined with other land-based measurements. As ocean

color is being used operationally by different agencies to report ocean conditions, there is a growing need to receive this type of data reliably and rapidly. These ever-increasing needs advocate for continuity in the deliverance of data products. Thus, not relying on a single complex satellite with a significant development time becomes attractive. The alternative of using a simplified satellite with lowered development time as a result of using Commercial-Off-The-Shelf (COTS) components (see chapter 2 or [5]), or better yet a constellation, become a viable option. Another necessary factor for operational use is the processing speed and reducing the delivery time from data acquisition to the processed product become available [2].

From a global perspective, ocean color technology has made significant contributions to our understanding of our environment for what should be considered a relatively modest investment. That is a relatively modest investment for larger nation or multi-nation organizations such as National Aeronautics and Space Administration (NASA) and ESA [2].

The large ocean area that makes up the Norwegian continental shelf can be difficult to monitor. While Norway is in a unique position concerning space technology and polar-orbiting satellites, Norway has not had the resources to launch many traditional satellites. The expenses and expertise needed to conduct traditional space missions are, after all, significant.

CubeSats for Ocean Color

A modern approach to space is through the use of CubeSats. The definition that formed the current Cube Satellite or CubeSat standard came around the year 2000. Smaller organizations, such as universities, can with this standard be able to develop applied novel space technology instruments and use-cases. The CubeSat standard states that the space crafts are built as a set of units (U) or cubes of $10\,\mathrm{cm} \times 10\,\mathrm{cm} \times 10\,\mathrm{cm}$. These CubeSats come in many shapes and formations from 0.25U to 24U, where the most frequently used form factor is the 3U CubeSat [18]. As the technology has matured, larger form factors like 6U and 12U have gained popularity. With the standard and increased maturity, the newer CubeSats are deployed with more advanced components such as more advanced payloads and deployable solar panels, which again leads to more advanced and ambitious missions [19].

This standardization helped create an emerging industry that provides a supply chain of CubeSat buses and subsystems. With this supply chain, the main focus of new and novel space applications is reduced to payload development, as the rest of the spacecraft bus can be procured from a CubeSat vendor. Thus, relying on the CubeSat stand can help ambitious space applications be realized at a reduced cost and development time in this new space paradigm. Advanced CubeSat missions have in recent years demonstrated that such platforms could implement low-cost science missions, with a considerable potential for a high return of investment in terms of scientific data and commercial value [20].

In recent years, CubeSats or other small satellite systems have provided an alternative for smaller groups to explore space-related topics. Earth observation is one space topic that has attained a lot of attention [20, 21], and the CubeSat platform is a popular choice for educational projects [21, 22]

1.3 HYPSO and Research Objectives

The MASSIVE project is an initiative at NTNU that seeks to research how to develop solutions to ocean monitoring in a manner that is effective both in terms of cost and time. The oceans make up most of the Earth and have a large impact on our climate, but as mentioned earlier, it is a challenge to monitor large and remote locations that are inaccessible for humans [2].

Sensor systems intended to provide ocean color remote sensing data collected by satellites and other airborne platforms that are enabling hyperspectral, rather than multispectral, data acquisition have been developed and tested at NTNU in recent years [5, 23, 24] The increased spectral and spatial resolution from these platforms is expected to enable optical detection of the pigment signature and its distribution [11]. Furthermore, the increased spectral resolution would enable the use of more advanced algorithms for the retrieval of parameters of interest, such as algorithms focusing on analytical known physical relationships, rather than empirical algorithms [7].

With other airborne sensors, like Unmanned Aerial Vehicles (UAVs) there will be the added ability to operate beneath a cloud cover, which can be a common occurrence along the Norwegian coast. Moreover, by augmenting the Ferrybox or similar Unmanned Surface Vehicles (USVs) with an imaging flow cytometer, it becomes possible to image single phytoplankton cells [11]. These images can then be transmitted directly to a forecasting center for algal bloom warnings and reduce the need for regular water-sample collection from observation sites of interest. Furthermore, Autonomous Underwater Vehicles (AUVs) and USVs can be equipped with optical sensors, and collect the data needed to verify and validate remotely sensed ocean color data from satellites used to derive parameters of interest [11, 24].

These new platforms and techniques that are being developed can enable a semi-autonomous network of observation platforms or agents. This can, in turn, provide real-time or near real-time data collection along the coast on potentially harmful species and pollution conditions before they reach hazardous concentrations. When new or even dangerous conditions are detected, the different autonomous or semi-autonomous agents can be directed to investigate further the areas of interest, guided by the overview provided by HYPSO-1 [23, 24].

The MASSIVE project intends to launch two small satellites designed to process hyperspectral data with novel on-board processing algorithms and efficiently distribute the processed information to other assets. The first of these satellites is the aforementioned HYPSO-1 satellite. The project also supports financing Ph.D. candidates and includes coordinating operations between satellites and other observation platforms such as autonomous AUVs and USVs. With the processed hyperspectral images from HYPSO-1, integrated with measurements obtained by other assets or satellites, this

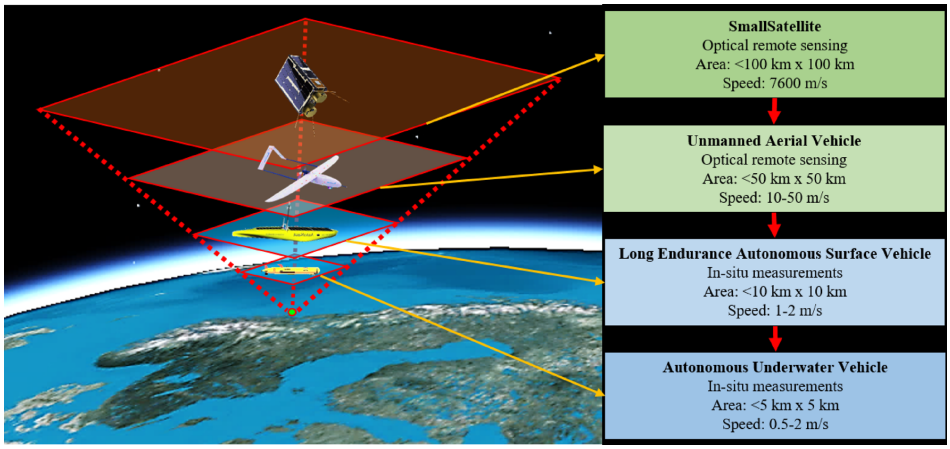


Figure 1.4.: The observational pyramid of the MASSIVE project. Figure from Mariusz E. Grøtte.

assimilated data is intended to, with appropriate processing, become useful information at different temporal and sptatial scales.

With the different capabilities of the various assets, the observation system becomes more flexible concerning changing weather conditions. It is important to recall that optical satellites, as the ones depicted in Figure 1.3, are not able to provide valuable data when there is high cloud coverage or dim light conditions [7]. Still, other assets, such as UAVs, could collect data despite the high cloud coverage, albeit with a reduced returning radiance signal due to reduced incoming light. With insight from weather forecasts, other ocean models, and other observation platforms, it is possible to better plan how to deploy the available assets. As discussed in [23, 24] this proposed system should be capable of providing target information from specific areas of interest in near real-time and near-continuous fashion.

This system of systems will target smaller areas. Thus, it is appropriate to support the calibration and validation activities using the other autonomous assets. With a fully integrated system of systems, the collected data can be used in a feedback loop. This data loop can be used to monitor the HSI performance and help make better and more accurate models for a given area over time.

1.3.1 The Knowledge Gap and Objectives

The research presented in this thesis focuses on how to develop a small satellite system with a flexible hyperspectral imaging payload and accompanying on-board processing for ocean color, with some of the challenges that a setting like a university environment can pose.

The overarching subject that the MASSIVE initiative addresses from the HYPSO perspective is to study how small satellite platforms that acquire hyperspectral image data can contribute valuable information to the ocean science community. Furthermore, we want to study how to maximize

information throughput by different processing strategies from this platform. This will augment ways to better assimilate and utilize information by integrating hyperspectral ocean color data with oceanographic models from other sensing systems in an observational pyramid. The observational pyramid envisioned is depicted in Figure 1.4. The Ph.D. research presented here aims to investigate potential venues for on-board processing, how to test and enable the payload in our small satellite system to be adaptable to changes in processing requirements in orbit, and how to facilitate the software development of such a payload in the given environment.

1.3.2 Research Questions

These objectives have formed the following three research questions.

RQ-1. How can a CubeSat platform be developed for hyperspectral ocean color applications and provide end-users with valuable information?

Hyperspectral image data from space is challenged by the available communication links regarding sending the vast data volumes that are acquired. This thesis presents a hyperspectral imaging CubeSat and accompanying processing strategies to reduce the data latency and deliver ocean color data products using advanced on-board processing.

RQ-2. What kind of algorithms or models for better hyperspectral data acquisition and reduced data latency can be deployed onboard a CubeSat tailored for ocean color observations?

As the ocean is a highly dynamic environment in the time and space domain, the data latency will affect the utility of the collected data. Different use cases for the data will further affect what kind of losses are acceptable. This thesis provides some strategies and perspectives on reducing information latency in various ways.

RQ-3. How can a CubeSat be used to support the development, validation, and upgrade-ability of new in-orbit data processing algorithms?

As the use of hyperspectral sensors from space for ocean color is a less explored scientific topic, it is helpful to adjust the algorithms used on-board during its operational phase. It is traditionally advisable not to attempt to do software upgrades in-orbit, but enabling it would extend the payload's capabilities beyond its initial launch configuration. This thesis presents a strategy for the software architecture design and testing of the payload that enables this functionality at an acceptable level of risk.

1.4 Thesis Outline and List of Publications

This thesis documents an introduction to some challenges of hyperspectral remote sensing for ocean color observations, CubeSat development, and Software development. This is not an extensive exploration of all there is to know about these complex topics but is intended to give an overview of the considerations taken into account when developing HYPSO-1.

The initial chapters 2 and 3 provides relevant details of the observation system under development. Chapter 4 concerns the use of a radiometer with the Skylight-Blocked Approach (SBA) to infer IOP from reflectance measurements and comparing them with data from HPLC and SpectroPhotometer (SP). This chapter demonstrates how established algorithms works in complex waters. Chapters 5, 6, 7, and 8 concerns different processing strategies that utilize the visual to near-infrared spectral range that can be acquired with the HYPSO-1 payload. More specific considerations with regards to the development of the payload is provided in chapters 9, 10, and 11, and concerns the development of a CubeSat in a University setting. These chapters provide the different lessons learned and documents the HYPSO teams attempt at accommodating the recommendations from space standards provided by organizations such as NASA and ESA, while tailoring the methodologies to our specific needs. Figure 1.5 provides an illustration of the overarching themes given in this thesis.

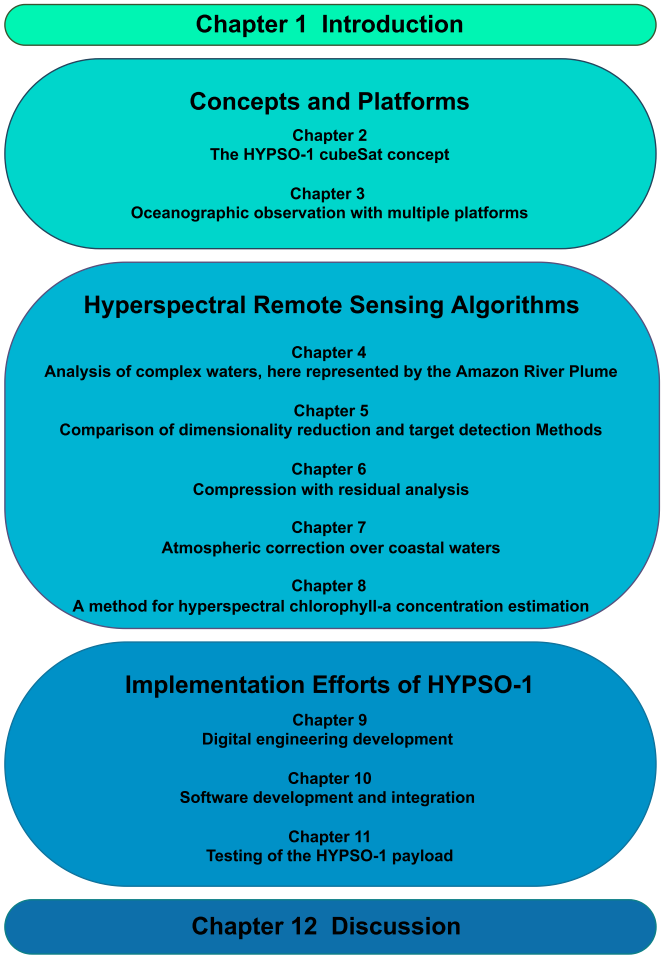


Figure 1.5.: An illustration of how the different chapters in this thesis is connected.

1.4.1 Included Contributions

The chapters within this thesis are adapted from publications I co-authored. Additional information stems from reflection and other related works. The publications are listed below in chronological order.

- S. Bakken, M. Orlandic, T. A. Johansen, **The effect of dimensionality reduction on signature-based target detection for hyperspectral remote sensing**, SPIE Optical Engineering + Applications Conference, CubeSats and SmallSats for Remote Sensing III, San Diego, 2019

  Adapted results found in Chapter 5. All authors conceptualized the paper. S.B. wrote, prepared the initial outline, and finalized the submission. S.B made the data acquisition, created the figures, and implemented the software. T.A.J and M.O. reviewed the draft and contributed to the final submission.

  This paper presents the effect of different methods for dimensionality reduction and noise removal on multiple classical methods for signature matched target detection often used in hyperspectral imaging, and suggests that dimensionality reduction can be combined with target detection for effective processing.

- O. M. Borge, S. Bakken, T. A. Johansen, **Atmospheric correction of hyperspectral data over coastal waters based on machine learning models**, 11th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2021

  Adapted results found in Chapter 7. O.M.B conceptualized the paper and prepared the initial outline and finalized the submission. O.M.B and S.B contributed to the discussion. All authors reviewed the draft and contributed to the final submission. The master thesis that formed the basis for this publication was awarded the Norwegian Industrial Forum for Space Activities (NIFRO) award for best master thesis in 2020.

  This paper presents different machine learning models for atmospheric correction of ocean color data, trained and evaluated using simulated hyperspectral ocean color data of top-of-the-atmosphere radiance from coastal waters to predict water-leaving radiance and other ocean color variables.

- S. Bakken, G. Johnsen, T. A. Johansen, **Analysis and model development of direct hyperspectral Chlorophyll-a estimation for remote sensing satellites**, 11th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2021

  Adapted results found in Chapter 8. S.B conceptualized the paper, prepared the initial outline, pared the figures, implemented the software, and finalized submission. T.A.J and G.J. reviewed the draft and contributed to the final submission.

This paper presents an analysis of a set of methods for estimation of [Chl a] from top of the atmosphere pseudo-reflectance using Partial Least Squares (PLS) and LASSO (Least Absolute Shrinkage and Selection Operator)-regression and compares the results from this with the internal consistency of the OC4 algorithm by NASA Ocean Biology Processing Group using data processed via SeaDAS 7.2 from the Hyperspectral Imager for the Coastal Ocean (HICO) mission.

- J. Garrett, S. Bakken, E. Prentice, D. Langer, F. Leira, E. Honoré-Livermore, R. Birkeland, M. Grøtte, T. A. Johansen, M. Orlandić, **Hyperspectral Image Processing Pipelines on Multiple Platforms for Coordinated Oceanographic Observation**, 11th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2021

  Adapted results found in Chapter 3. All authors conceptualized the paper. J.G. wrote and prepared the original draft.

  This paper presents how the Norwegian University of Science and Technology is planning to build an ocean observation system, including HYPSO, to monitor ocean color in coastal waters with an ensemble of robotic agents.

- M. E. Grøtte, R. Birkeland, E. Honoré-Livermore, S. Bakken, J. L. Garrett, E. F. Prentice, F. Sigernes, M. Orlandic, J. T. Gravdahl, T. A. Johansen, **Ocean Color Hyperspectral Remote Sensing with High Resolution and Low Latency - the HYPSO-1 CubeSat Mission**, IEEE Trans. Geoscience and Remote Sensing, 2021

  Adapted results found in Chapter 2. All authors conceptualized the paper. J.L.G. and S.B. wrote and prepared the initial outline and finalized the section on on-board image processing architecture, T.A.J and M.O. reviewed the draft and contributed to final submission.

  This paper present the mission design for HYPSO, as well as different concepts with regards to the FPGA-based on-board image processing architecture that aim to reduce the data volume without sacrefising important spatial-spectral information.

- E. Honoré-Livermore, R. Birkeland, S. Bakken, J. L. Garrett, C. Haskins, **Digital Engineering Management in an Academic CubeSat Project**, Accepted for publication in Special Issue on Systems Engineering Challenges in Journal of Aerospace Information Systems (2021)

  Adapted results found in Chapter 9. All authors conceptualized the paper. E.H-L. wrote and prepared the initial outline and finalized the submission. S.B., R.B. and J.L.G. all contributed to the background, results, and discussion sections. C.H. reviewed the draft and contributed to the final submission.

  This paper presents how digital engineering can be used in an academic CubeSat project, such as HYPSO, in which a variety of techniques for workflow, on-boarding, and communication

are tested, and the tailoring that has been applied to fit the academic environment is described with promising results.

- S. Bakken, E. Honoré-Livermore, R. Birkeland, M. Orlandić, E. Prentice, J. Garrett, D. Langer, C. Haskins, T. A. Johansen, **Software Development and Integration of a Hyperspectral Imaging Payload for HYPSO-1**, 2022 IEEE/SICE International Symposium on System Integration (SII), 2022, Accepted.

  Adapted results found in Chapter 10. S.B conceptualized the paper and prepared the initial outline and finalized the submission. S.B, E.H.L and R.B analyzed the results. All authors reviewed the draft and contributed to the final submission.

  This paper presents the software architecture, development, and integration of a COTS based hyperspectral imaging payload on-board the HYPSO-1 CubeSat.

- S. Bakken, J. Garrett, R. Birkeland, M. Orlandić, E. Honoré-Livermore, P. A. R. Marton D. Langer, C. Haskins, T. A. Johansen, **Testing of Software-Intensive Hyperspectral Imaging Payload for the HYPSO-1 CubeSat**, 2022 IEEE/SICE International Symposium on System Integration (SII), 2022, Accepted.

  Adapted results found in Chapter 11. R.B and S.B conceptualized the paper. S.B prepared the initial outline and finalized the submission. S.B, J.G, P.A.R.M and R.B contributed to the discussion. All authors reviewed the draft and contributed to the final submission.

  The importance of embedded software for CubeSats flying COTS based payload systems has increased to provide more functionality and flexibility, but these payload systems warrant extensive software and hardware testing. This paper presents the ongoing work on how the payload software testing is done for HYPSO-1.

- S. Bakken, P. Rossvoll, T. Pedersen, M. Orlandic,T. A. Johansen, and H. Martens, **Compression by PCA and JPEG2000 with Residual Analysis for Hyperspectral Remote Sensing**, IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, **Submitted**

  Adapted results can be found in Chapter 6, H.M. conceived the idea; S.B and M.O. developed the methodology; S.B performed the implementations, validation, formal analysis, data processing and prepared the original draft; S.B and M.O performed the analysis and visualization; P.R, T.P, M.O., T.A.J., and H.M. contributed to the investigation, interpretation of the results, writing-review and editing; T.A.J. supervised the work

  This paper shows how residuals from transformation-based lossy compression by pre-computed Principal Component Analysis (PCA) can be used to improve the pre-computed transformation matrix, provide insight into the limitations of the transformation, and provide greater confidence in data that is retrieved using transformation-based encoding as a lossy encoding scheme.

- Sivert Bakken, Geir Johnsen, Tor Arne Johansen, Jianwei Wei, Zhongping Lee, Joseph Montoya, and Ajit Subramaniam, **Hyperspectral Remote Sensing and Analysis of the Amazon River Plume**, TBD, **Submitted**

  Adapted results found in Chapter 4. S.B and A.S. developed the methodology mentioned as a contribution; J.W, Z.L loaned the SBA HyperPro instrument and provided guidance on deployment and initial analysis of the data acquired; JPM helped with station locations and deployments; S.B performed the implementations, validation, formal analysis, data processing, and analysis, visualization, and prepared the original draft; A.S, G.J, T.A.J., and A.S. contributed to the investigation, interpretation of the results, writing-review, and editing; A.S. supervised the work;

  This paper shows our initial results from capturing hyperspectral data using the SBA-based approach with a radiometer in the Amazon River plume. The results are compared with data from HPLC-analysis and SP.

1.4.2 Other Contributions

The publications I co-authored but did not include, are listed below in chronological order.

- *M. Lapadatu, S. Bakken, M. E. Grøtte, M. Alver, T. A. Johansen,* ***Simulation tool for hyperspectral imaging from a satellite****, 10th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Amsterdam, 2019.*

- *D. Bošković, M. Orlandić, S. Bakken and T. A. Johansen,* ***HW/SW Implementation of Hyperspectral Target Detection Algorithm****, 8th Mediterranean Conference on Embedded Computing – MECO, Montenegro, 2019.*

  The master thesis that formed the basis for this publication was awarded the NIFRO award for best master thesis in 2019. The submitted paper was awarded a prize for the 2nd best student paper at the MECO conference.

- *S. Bakken, J. Wei, Z. Lee, G. Johnsen, T. A. Johansen, J. Montoya, A. Subramaniam, Hyperspectral Remote Sensing of the Amazon River Plume, Ocean Sciences Meeting, 2020*

- *Honoré-Livermore, Evelyn; Bakken, Sivert; Prentice, Elizabeth Frances,* ***Factors Influencing the Development Time from TRL4 to TRL8 for CubeSat Subsystems at a University****, 9th International Systems & Concurrent Engineering for Space Applications Conference (2020) (virtual)*

- *A. Oudijk, F. Sigernes, H. Mulders, S. Bakken, T. A. Johansen,* ***Quality assessment of standard video compression techniques applied to hyperspectral data cubes****, 11th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2021*

Chapter 2

The HYPSO-1 CubeSat Concept

Far and away, the greatest threat to the ocean,
and thus to ourselves, is ignorance.
But we can do something about that.

Sylvia Earle

Sporadic ocean color events with characteristic spectra, in particular algal blooms, call for quick delivery of high-resolution remote sensing data for further analysis. This chapter present the mission design of HYPSO-1, a 6U CubeSat at 500 km orbit altitude hosting a custom-built pushbroom hyperspectral imager with wavelengths between $387 - 801$ nm at bandpass of 3.33 nm and swath width of 70 km. The expected Signal-to-Noise Ratio is characterized for typical open ocean water-leaving radiance and can be flexibly increased by pixel binning. The HYPSO-1 CubeSat was successfully launched on the 13th of January 2022 and is being commissioned and operating as expected. Since generated high-dimensional hyperspectral data products need to be transmitted over limited space-ground communications, we have designed a modular FPGA-based on-board image processing architecture to reduce the data size without losing important spatial-spectral information. We justify the concept with a simulated scenario where HYPSO-1 first collects hyperspectral images of a 40 km by 40 km coastal area in Norway, and aims to immediately transfer these to nearby ground stations. With CCSDS123v1 lossless compression, it takes about one orbit revolution to obtain the complete data product when considering the overhead in satellite bus communications, and less than 10 min without the overhead. It is discussed how better latency can be achieved with advanced on-board processing algorithms.

Hyperspectral and multispectral remote sensing are typically used in the context of monitoring colorful processes with large Spatio-temporal extents. A commonly observed phenomenon of these is chlorophyll, a primary light-absorbing substance involved in phytoplankton photosynthesis which may have clear signatures at the water surface [25]. Blooms of phytoplankton have variable coloration and are often categorized as “red tides”, “green tides” or “brown tides” with wavelengths

between $400 - 700$ nm [13, 25–29]. They sporadically appear worldwide with varying biomass concentrations, may last from a few minutes to several days and cover regions from tens to hundreds of square kilometers [30]. Sometimes malignant blooms, often identified as Harmful Algal Blooms (HABs), may cause sudden damage to the marine environment, ecosystems, and sustainable food sources [31]. According to [32], numerous plankton and algae types can be distinguished or inferred by their photosynthetic pigments and fluorescence, and hyperspectral data with the capability of high spectral resolution may reveal the subtle spectral inflections imparted by specific pigment complements. However, determining the harmfulness of algae is not done from optical remote sensing alone and is attributed to in-situ measurements in the upper water-column [26, 33]. Further challenges include atmospheric absorption, and scattering of light [34], and the fact that the majority of biomatter typically reside at $10 - 15$ m below the water surface [26], such that these heterogeneous and potentially dark targets often demand a combination of larger space-based optics with a high Signal-to-Noise ratio (SNR), rigorous atmospheric correction schemes and accompanied real-time in-situ measurements [35, 36].

Traditional Earth Observation (EO) satellites with large optical systems, several operated by National Aeronautics and Astronautics Administration (NASA) and European Space Agency (ESA), are designed to cover the Earth on a global scale and provide excellent ocean color data with medium to high spatial resolution [37, 38]. However, they usually offer low spectral resolution and revisit times of several days [39]. For example, using data products from Sentinel-3's Ocean Color and Land Instrument (OLCI) for detecting cyanobacterial blooms based on pigments such as phycocyanin and chlorophyll-a can be inaccurate using traditional ground-based analysis algorithms unless employing newer algorithms that utilize band ratios from an alternate set of selected spectral bands [40]. Providing greater flexibility in namely choosing more than a hundred spectral bands instead of dozens [41], hyperspectral remote sensing missions show great promise in ocean color remote sensing, e.g. [8, 42–50]. Nevertheless, many of these stand-alone systems still lack the operational flexibility and revisit times to monitor dynamic areas on-demand [51] efficiently. Moreover, accurate but time-consuming and rigorous data processing methods are usually performed on the ground together with synergistic analysis of in-situ measurements.

Instead of mapping on the global scale, single-purpose hyperspectral imaging small-satellites may focus on observing smaller dedicated areas more frequently to characterize temporal variation in both the spatial and spectral domains, also allowing a miniaturized camera system with relatively narrow Field-Of-View (FOV). Choosing target areas on the sub-mesoscale or mesoscale, has the potential to enable small satellites to support a network of in-situ assets that may observe or sample with more detail in the spatial and spectral domains, e.g., Unmanned Aerial Vehicles (UAVs), Unmanned Surface Vehicles (USVs), Autonomous Underwater Vehicles (AUVs), and buoys [31]. To make such a multi-agent network function efficiently in real-time and reduce the operational costs, the remote sensing data must be quickly downloaded to keep validity in the highly time-varying information, as discussed in chapter 3.

It is well-known that hyperspectral imagers generate large amounts of data that consequently take a long time to transfer to the ground due to limitations in bandwidth, coverage to ground stations and on-board computational resources [51, 52]. Reduction in data size on-board is crucial for satisfying real-time requirements but can also be difficult due to the limited power available per orbit for a small satellite. Nevertheless, on-board processing has advanced significantly for remote sensing applications [53], in particular, Field-Programmable Gate Arrays (FPGAs) that are reconfigurable and have high computational speed and low power consumption [54, 55]. Enabling algorithm parallelism, a modular FPGA-based image processing architecture, allows for custom algorithms or image processing pipelines. Beyond standard losslessly compressed data, tailored end data products may contain only extracted spatial-spectral information generated from dimensionality reduction, target detection, or classification [56, 57]. The significantly reduced data can therefore grant a shorter waiting time between image acquisition to complete data download and be used for immediate utilization in real-time applications, e.g., for in-situ measurements or supporting an algal bloom warning systems [11, 29, 58, 59].

This chapter presents the mission design for the upcoming HYPer-spectral Smallsat for ocean Observation (HYPSO-1) developed at the Norwegian University of Science and Technology (NTNU). This Smallsat is designed to support environmental monitoring and the ocean color community by providing customized hyperspectral data products with low latency. This chapter is adapted from [23]. Section 2.1 describes the ocean color remote sensing needs that motivate the choice of an imager, key remote sensing capabilities, and the HYPSO-1 Concept of Operations (CONOPS).

Section 2.2 describes the envisioned FPGA-based on-board image processing pipelines that shall deliver custom data products, provide a survey of potential on-board implementations of more advanced algorithms, and justify the HYPSO-1 mission feasibility with corresponding data latency for chosen imaging modes and the user-attuned data products. Finally, some conclusions for this concept is provided in Section 2.3.

The design and performance of the custom pushbroom hyperspectral imaging payload is covered in [5, 23], and the proposed remote sensing strategy, supported by results from simulations, using a slew maneuver to enhance the spatial resolution in the image is discussed in [23]. The HYPSO-1 system is described in Appendix A.

2.1 Mission Design

This is section presents the mission requirements of the HYPSO-1 system. The following sections attempt to relate these requirements and their reasoning regarding what the software shall enable the system to do. chapter 10 provides further details on how the software architecture was designed to support the mission design.

2.1.1 Objectives

The mission objectives of the HYPSO-1 are to monitor the spatio-temporal extent of ocean color events in the visual and near-infrared (VIS-NIR) wavelengths between $400 - 800$ nm; and attempt to infer phytoplankton functional groups. Key user needs for ocean color remote sensing are:

1. Images should have spatial resolution better than $30 - 100$ m per pixel [39, 60];
2. Raw hyperspectral data should have spectral resolution of about 5 nm for VIS-NIR wavelengths [39, 60];
3. The imager's SNR at Top of Atmosphere (ToA) should be greater than 400 in visual wavelengths for open ocean water [61], and atmospherically corrected SNR of water-leaving signals should be between $40 - 100$ [62];
4. Data latency should be less than 1 hr [63];
5. Revisit times to dedicated areas of interest should be $3 - 72$ hrs [63, 64].

Since HYPSO-1 is a single small-satellite, but the first in a prospective constellation, we focus on working towards the recommendations 1), 2), 3) and 4).

2.1.2 Image Acquisition Basics

Whereas several types of spectrometers can be integrated on aerial or space platforms [65], the passive pushbroom imager design is an attractive choice with good SNR [4,66,67]. The use of COTS components has also made this type of design more affordable, accessible, and flexible [68, 69].

With the scan direction oriented towards the velocity direction, a pushbroom imager sequentially scans several lines, N_x, each having instantaneous pixels, N_y and N_λ, forming a hyperspectral datacube shown in Figure 2.1. N_y is the number of spatial pixels perpendicular to the scan direction, and N_λ is the number of pixels along the spectral dimension. The horizontal and vertical components of the FOV are ϵ_w and ϵ_h, respectively. The time elapsed between two consecutive lines, or frames, is expressed by the integration time $\Delta t = 1/FPS = \tau + \delta t$ where FPS is the frames per second or frame rate, τ is the camera exposure time, and δt is the read-out time.

A high spectral resolution is required to discriminate fine spectral features in the water-leaving signals, and high spatial resolution is desired to reduce the effects of spectral mixing or blur in the image pixels. Mounted on a satellite moving at high orbital speed, the drawback is, strictly speaking, a low spatial resolution along the scan direction. A workaround is to overlap more frames by tilting the imager backward as it translates forward, similar to the method described in [70]. This results in an increased number of partially overlapping pixels which can be utilized to enhance SNR or spatial resolution as trade-offs using image restoration techniques such as deconvolution or super-resolution [71]. For clarity, the Euclidean distance on ground between center pixels of two consecutive frames is taken to be the Sequential Ground Sampling Distance (SGSD) not to be

confused with the commonly defined Ground Sampling Distance (GSD) between adjacent pixels in an instantaneous frame.

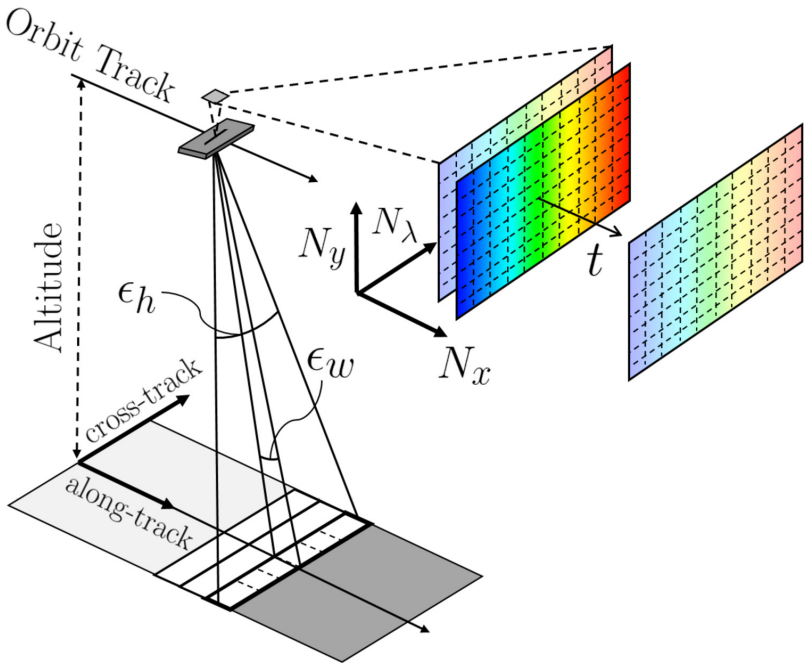


Figure 2.1.: Illustration of a pushbroom hyperspectral imager collecting N_x frames with N_λ and N_y pixels. Provided by Mariusz E. Grøtte.

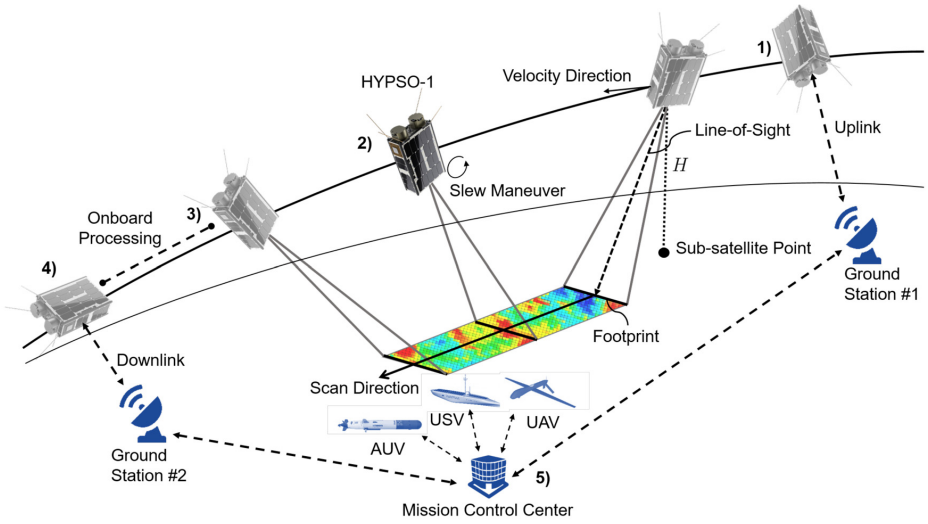


Figure 2.2.: CONOPS where 1) HYPSO-1 receives uplinked configurations from a nearby ground station; 2) acquires hyperspectral images for a short duration under a slew maneuver; 3) processes the images onboard immediately; 4) downlinks the data to nearby ground stations; and 5) in-situ assets in the vicinity may be deployed for closer investigation at the observed scene. Provided by Mariusz E. Grøtte.

2.1.3 Concept of Operations

The overall mission utility and performance in HYPSO-1 is mainly engineered based on trade-offs in spatial resolution, spectral resolution, SNR, data size and latency, coverage to ground stations and likely locations for observations. HYPSO-1 will be launched to a 500 km altitude Sun-Synchronous Orbit (SSO) with Local Time of Descending Node (LTDN) at 10:00 AM, which grants early-day access to observe the Norwegian coastline during Spring and Summer seasons while avoiding detrimental sun-glint effects [72]. The HYPSO-1 mission concept of operations (CONOPS), illustrated in Figure 2.2, enables five main capabilities:

1. After receiving telecommands and updates (e.g., camera settings) that are uploaded from a nearby ground station, HYPSO-1 is scheduled to orient its hyperspectral imager to start scanning a pre-defined area size;
2. HYPSO-1 executes a single-axis slew maneuver so that the imager's footprint slowly rotates backwards with respect to the scan direction. At a high camera frame rate, the goal is to enable a SGSD better than 100 m/pixel.
3. After imaging, the hyperspectral images are processed onboard immediately to reduce their data size and to speed up the download to the ground;
4. For quick downlink after observing coastal regions in Norway, the selected ground station network includes S-band ground stations at NTNU Trondheim, KSAT Svalbard, Norway, and KSAT Puertollano, Spain;
5. In addition, the Mission Control Center at NTNU operates several supporting robotic assets, such as UAVs, ASVs and AUVs, that may collect in-situ data if within range of the observed area.

2.1.4 System Capabilities

Imaging Modes

The hyperspectral imager has three main imaging configurations:

- High-resolution mode: enables high image resolution with narrow-FOV and high frame rate settings;
- Wide FOV mode: enables a wider swath but at coarser spatial resolution;
- Diagnostics mode: gives raw data with full sensor resolution mainly for in-orbit calibration and characterization.

Attitude Determination & Control System

To obtain a spatial resolution better than 100 m requires a precise attitude determination and control system (ADCS) [73]. Throughout image acquisition for a satellite that is pointing or maneuvering, the attitude sensor noise and actuator inaccuracies (e.g. reaction wheel jittering) will contribute to a non-uniform distribution of images across the observed scene. The attitude errors are categorized as attitude control and knowledge accuracies, bearing in mind that the latter's performance affects the former. For consistent image registration, or simply knowing the location of each pixel to the accuracy of 100 m on the ground, e.g. geo-referencing, then good performance is needed for attitude knowledge accuracy, orbit position accuracy, and time synchronization between the captured images and attitude data.

On-board Image Processing

The image processing architecture should be modular by design to ease satellite operations and provide tailored data to end-users at a low data latency. To make such data products useful, the high-level goals are to:

- Reduce hyperspectral data size onboard to improve data latency, by lossless compression at a minimum;
- Extract the spatial and spectral information in water-leaving signals, by e.g. dimensionality reduction, target detection or classification;
- Register images and utilize the obtained SGSD to achieve better than 100 m/pixel image resolution using image restoration methods, e.g. deconvolution or super-resolution;
- Transform pixel indices to geodetic latitude and longitude by geo-referencing such that these coordinates can be used to guide in-situ agents towards interesting locations;

The hyperspectral data products shall be analyzed in synergy with other available remote sensing data and in-situ measurements in the commissioning phase. Modeling and simulation tools shall also provide supporting information on atmospheric correction and the radiometric, spectral and spatial properties of a simulated ocean color event [74, 75].

2.2 On-Board Image Processing Architecture

The FPGA-based image processing algorithms on the OPU are essential to enable faster download and distribution of data while at the same time relieving HYPSO-1's power budget. The idea behind the image processing architecture is to allow for modular arrangement of algorithms, or pipelines, as illustrated in Figure 2.3. The minimal, dimensionality reduction, target detection, and classification on-board image processing pipelines (respectively named MOBIP, DROBIP, TOBIP and COBIP) are designed to generate customized data products depending on the particular need of the user or operator. All pipelines include image acquisition, time-stamping and binning prior to image processing. It is also critical that satellite and payload telemetry and any other relevant

metadata are downlinked together with the processed images, including ADCS and orbit position data collected during image acquisition. Table 2.1 shows the size reduction and processing speeds for the suggested algorithms to be employed in the architecture. "Bands/Components" are referred to as spectral bands for raw data and MOBIP, extracted components for DROBIP, a probability map of detected target spectral signatures for TOBIP, and a layer containing classes of spectra for COBIP. A raw hyperspectral datacube of 956×684 spatial pixels and 1080 spectral pixels binned by a factor of $B_\lambda = 9$ times is considered the starting point for further processing. The data size reduction and processing speed estimates are based on performance reported on state-of-the-art image processing algorithms that have been used on hyperspectral data of similar sizes. Details related to occupation, execution time, operating frequency, and latency of the following FPGA-based algorithms can be found in the respective literature on their implementation.

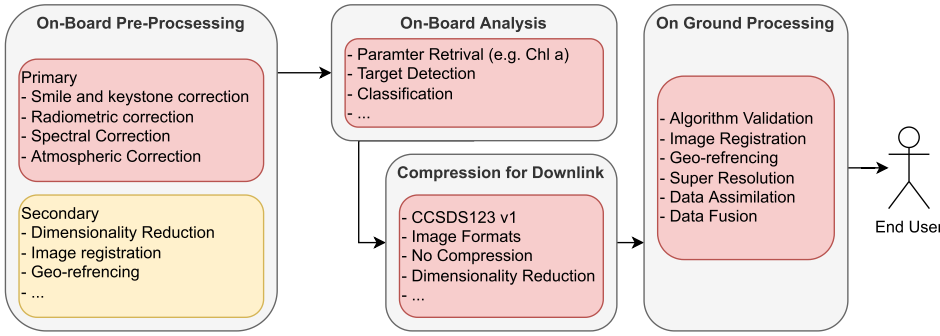


Figure 2.3.: Block diagram of the proposed imaging processing pipelines. The hyperspectral images are captured, binned, processed at chosen level, stored on SD-card and downlinked together with collected telemetry and metadata. Depending on the downlinked data product, additional ground-based processing and fine-tuning can be applied before distribution to the end-user. In-situ validation and data fusion with other remote sensing data are critical aspects of the HYPSO-1 data validation. Black arrows indicate the minimal on-board processing pipeline while gray arrows are the alternative routes for tailored data products.

Table 2.1.: Hyperspectral Data Products for $N_x \times N_y = 956 \times 684$ spatial pixels

Level	Bands	Bit Depth	Total (MB)	Reduction (%)	Speed (Mbps)
Raw Binned (at $B_\lambda = 9$)	120	16	156.94	-	-
CCSDS123 (MOBIP)	120	16	69.75	55.6	6260^a
PCA (DROBIP)	20	16	11.89	92.4	268.6^b
OTFP (DROBIP)	20	16	11.89	92.4	7^c
EMSC (DROBIP)	20	16	11.89	92.4	54.5^d
Target detection (TOBIP)	1	16	1.308	99.2	903.1^d
Classification - 16 classes (COBIP)	1	4	0.331	99.8	53.2^c
Classification - 256 classes (COBIP)	1	8	0.711	99.6	53.2^c

a : measured on Zynq-7030 Xilinx FPGA flight hardware on OPU.
b : estimated from Xilinx Virtex-7 XC7VX690T FPGA-based software/hardware co-design tests on similar hyperspectral data size.
c : estimated from software implementation test on similar hyperspectral data size.
d : estimated from Zynq-7000 FPGA-based software/hardware co-design tests on similar hyperspectral data size.

2.2.1 Minimal On-Board Image Processing

MOBIP consists of the CCSDS123v1 lossless compression algorithm [76], applied after image acquisition, time-stamping and binning. CCSDS123v1 proposed in [77, 78] offers a measured FPGA core speed of up to 9984 Mbps on HYPSO-1's Zynq-7030 Xilinx PicoZed. With the raw binned image as a starting point, data compression of at least 55.6% is measured as shown in Table 2.1, i.e. a reduction factor of 2.25 times. Without spatial or spectral information loss, any user can independently process and analyze this data product on the ground. Furthermore, the CubeDMA, a Direct Memory Access (DMA) solution, is built in the FPGA to ensure an efficient stream of hyperspectral images by excluding the Central Processing Unit (CPU) from its critical path of transfer and establishing direct communication between the memory and the dedicated CCSDS123v1 processing core [79].

2.2.2 On-Board Image Processing for Tailored Data

Given the in-orbit reconfigurability of the FPGA, several suitable algorithms that can be used in DROBIP, TOBIP, COBIP are described here. Some are demonstrated as FPGA-implementations or software/hardware co-design in relevant hardware, and a few algorithms run in software that needs further development for optimized implementations to run on-board.

The results from chapters 7 and 8 are not included here, but would have a similar data volume as the TOBIP in table 2.1, with an additional band if one were to add an confidence estimate or similar. The results from 6 are not directly represented in the table either.

Dimensionality reduction

Dimensionality reduction methods extract the main spectral patterns and remove redundancies from the high-dimensional hyperspectral data. Applying dimensionality reduction as a pre-processing step before any succeeding algorithms increases overall computational efficiency [80], and the practical spatial-spectral features of interest can be used, e.g., studying the water-leaving radiance and atmospheric effects in an observed heterogeneous scene. As shown in Table 2.1, with 20 components chosen, a size reduction rate is estimated to be 92.4% when combined with CCSDS123v1. An optional step before dimensionality reduction can be to apply smile and keystone corrections to prevent intertwining systematic artifacts irrevocably in the data by adjusting the images to account for systematic optical and measurement noise inherent to the hyperspectral imager [81].

A common dimensionality reduction technique is Principal Component Analysis (PCA) which obtains a reduced and de-noised subspace representation of the raw hyperspectral data, assuming a linear model with Gaussian noise [82]. The extracted spatial-spectral information in a scene are contained in only a few principal components instead of a hundred spectral bands. An FPGA implementation of PCA in Xilinx Virtex-7 XC7VX690T proposed in [83] is reported to obtain computational speed of 4.17 s when used to extract 24 principal components from an Airborne

visible/infrared imaging spectrometer (AVIRIS) image of Jasper Ridge Biological Preserve, California, with 614×512 spatial pixels and 224 spectral channels, and is fast enough to process a stream of hyperspectral images in real-time. According to [84], an adaptive bilinear PCA-based On-the-Fly Processing (OTFP) algorithm may sequentially process streaming blocks of data instead of analyzing the whole dataset at the end of image acquisition. Although in Matlab, its reported computational speed is 300.2 s for obtaining three principal components from a 16-bit hyperspectral image of 1000×245 spatial pixels and 450 spectral channels, however higher speed is expected for an FPGA implementation. An alternative method to PCA, the Extended Multiplicative Signal Correction (EMSC), estimates a de-noised subset of relevant spectra using a linear statistical model of observations with approximated light absorbance and scattering [85]. A software/hardware co-design of EMSC on a Zedboard development platform with ARM Cortex-A9 processor measures a computational time of 3.81 seconds when applied on a 16-bit hyperspectral datacube with 500×500 spatial pixels and 50 spectral channels.

Target Detection

Hyperspectral images of heterogeneous scenes are amenable to spectral-based target detection because of their numerous spectral bands [86, 87]. Effective use of target detection in hyperspectral imagery requires a set of a-priori known target spectra, and high spatial resolution is desired to reduce the effects of spectral mixing in the spatial pixels. Target detection generates a probability map of target spectral signatures across the image in the spatial domain, resulting in a two-dimensional data product per chosen number of signatures as indicated in Table 2.1. As an example, only one target signature is chosen such that the size of the two-dimensional map is $1 \times 956 \times 684 \times 16$ bits $= 1.308$ MB, i.e. a size reduction of approximately 99.2% of the original data. Due to the small data size, the reduction for target detection in Table 2.1 is assumed to not include lossless compression with CCSDS123v1.

Proposed in [88], the target detection module supports Constrained Energy Minimization (CEM), Adjusted Spectral Matched Filter (ASMF) and modified Adaptive Cosine Estimator (ACE) detectors to determine the likelihood of specific spectral signatures in a spatial pixel. For real-time computation on a stream of hyperspectral images, dimensionality reduction should be applied as a pre-processing step. For software/hardware co-design of modified ACE algorithm on a Zedboard development platform with ARM Cortex-A9 processor, the computational time is reported to be 3.29 s for an input of HyMap 16-bit hyperspectral datacube of 224000 spatial pixels and 16 principal components given PCA pre-processing [89]. A computational time of 0.5 s is reported for FPGA-implementation of modified ACE algorithm on a Zynq-7035 SoC (Kintex-7) applied on the complete HyMap datacube with 126 spectral bands without PCA pre-processing [88], which is used as benchmark estimate in Table 2.1.

Classification

Using a spatial-spectral classification framework, the spatial pixels in a hyperspectral image can be separated into different classes based on spectral signatures [90]. One of many such classification techniques that are suitable for FPGA-implementation is the Fast Spectral Clustering (FSC), a graph-based unsupervised method that does not require training data [91, 92]. Indicated in Table 2.1, it is possible to represent each pixel or layer with a 4 bit integer for fewer than 16 classes, whereas 256 classes can be represented with 8 bits. The size of 16 class signatures with 120 spectral bands per signature is $16 \times 120 \times 16$ bits $= 0.0038$ MB and for 256 class signatures the size is $256 \times 120 \times 16$ bits $= 0.06144$ MB. These auxiliary data products are added to the classification map with size of $1 \times 956 \times 684 \times 4$ bits $= 0.327$ MB for 16 classes and $1 \times 956 \times 684 \times 8$ bits $= 0.654$ MB for 256 classes. The estimated data size reduction is 99.8% and 99.6%, respectively. Due to the small data sizes, the reductions for classification in Table 2.1 are assumed to not include lossless compression with CCSDS123v1.

The computational speed for a Nystrøm Extension Clustering version of FSC, described in [91], is estimated to be about 245.8 Mbps based on a Matlab implementation that resulted in 1.62 s used for providing 16 classes from a 16-bit AVIRIS image of Salinas Valley, California, with 512×217 pixels and 224 spectral bands. Even higher speed is expected for a software/hardware co-design of FSC on FPGA. An alternative to FSC is the potentially more accurate Clustering using Binary Partition Trees (CLUS-BPT) framework, which integrates embedded hyperspectral data segmentation, region modeling, feature extraction by PCA and clustering [93]. Its reported computation time in Matlab is 7.48 s for the same AVIRIS image. However, FSC generally outperforms the CLUS-BPT in computational time for an input image with large spatial dimensions. As a worst-case in Table 2.1, the estimated processing speed is therefore assumed to be 53.2 Mbps based on CLUS-BPT. Naturally, as with any other on-board processing algorithms, cropping the images in the spatial domain to focus on specific regions will improve the processing latency in classification.

2.2.3 Discussion on Advanced Algorithms

Given the FPGA reconfigurability, other relevant algorithms beyond those discussed may be uploaded to the OPU in-orbit if potential maturity is reached. Generally, the algorithms should first be rigorously tested on the ground with careful validation of processing characteristics such as speed, reliability, and acceptable accuracy resulting in the data. Relevant algorithms include image registration, geo-referencing, atmospheric correction, and super-resolution, which may improve target detection and classification accuracy. However, these algorithms are generally too computationally intensive and complex for on-board implementation and real-time use. Further studies, development, and testing are required.

Image Registration

Image registration determines relative separation between individual pixels, sometimes named orthorectification. Such algorithms are in general, too computationally expensive for on-board applications [94]. A more straightforward ray-tracing method can be adapted for on-board implementation, which has been prototyped for joint registration and geo-referencing, similar to the one described in [95].

Geo-referencing

The benefit of on-board geo-referencing lies in directly downlinking the latitude and longitude coordinates of pixels with target detection or classification results. This requires direct inputs of time-synchronized ADCS and navigation data. For ground use, instead of downlinking the whole data product from, e.g., target detection or classification, it is possible to transfer only the relevant spatial pixel indices to be geo-referenced. This results in much smaller data size and latency, and in-situ agents nearby HYPSO-1's observed area can, therefore, quickly be commanded to travel to these coordinates for closer inspection.

Super-Resolution

Super-resolution algorithms can be adapted to enhance the spatial resolution in images as described in [96], and thereby improve the radiometric and geometric accuracy. Super-resolution prototypes require a measurement process, e.g. determining the point-spread function, to infer higher spatial resolution in the image [97, 98], which are based on methods from multi-frame super-resolution [99, 100]. Although these types of algorithms can improve the spatial image resolution, they are susceptible to noise, quantization, compression, and inaccuracies in the estimate of the point spread function [101, 102]. Prior-based super-resolution techniques, such as sparse image representations [103] and convolutional neural nets [104, 105], namely overcome the limitations in measurement-based techniques by supplementing input pixels with expectations of hyperspectral image statistics. Other methods involve using multispectral-hyperspectral image fusion [106, 107] and super-resolution based on dimensionality reduction [108].

An FPGA-implementation of a Richardson-Lucy (RL) deconvolution algorithm on Xilinx Zynq-7020 Zedboard with two ARM Cortex-A9 cores proposed in [109], has been successfully applied on hyperspectral data, where a computational time of 1.06 ms is reported per iteration when processing a band with a size of 150×640 pixels by using kernel size of 9×9 pixels. Corresponding software/hardware co-design of the deconvolution algorithm is proposed in [110].

Atmospheric Correction

Removing atmospheric effects before dimensionality reduction, target detection or classification, can improve the accuracy and efficiency in extracting or detecting relevant water-leaving signals. The purpose of atmospheric correction is to identify the terms in the top-of-the-atmosphere radiance that contribute to the total radiance and predict the actual water-leaving radiance component,

which may further contain the optical properties of important ocean color parameters such as chlorophyll.

Many ground-based atmospheric correction schemes work well for open ocean waters for multi-spectral data [9, 111–113], and good performance has also been shown for hyperspectral images of coastal waters [114]. Traditional atmospheric correction methods are generally built on the radiative transfer model [4], but they are not designed for on-board real-time applications due to the complexity and computational expense. Without contemporary empirical or ground truth data, they can also be prone to over-or under-correction of the radiance terms, resulting in significant radiometric inaccuracies for highly variable coastal waters and an unpredictable atmosphere. On the other hand, effective non-deterministic atmospheric correction methods using machine learning, e.g., neural networks, have regularly been employed and are considered to be robust given a proper set of training data [2]. If hyperspectral images and ground truth data are unavailable for training, simulation tools such as Accurate Radiative Transfer (AccuRT) [115], based on a coupled atmosphere-ocean radiative transfer model, could simulate heterogeneous scenes of complex water and atmospheric conditions. This simulations study is presented in chapter 7. A suitable on-board FPGA implementation of atmospheric correction methods still needs further investigation.

2.2.4 Dynamic Reconfiguration

Using FPGAs can overcome the limited hardware resources on-board a small satellite and the increasing performance requirements for on-board processing in terms of processing complexity and spatial-spectral resolution in hyperspectral data. Using Dynamic Reconfiguration (DR) on FPGAs, reconfigurable solutions obtain the needed flexibility and allow changes and adaptation of the on-board processing. In HYPSO-1, the DR can increase resource utilization by switching between different processing pipelines and for functional updates and upgrades in each pipeline that are uplinked from the ground. An advanced ability of modern FPGAs is Dynamic Partial Reconfiguration (DPR) that reprograms portions of the FPGA, while the rest of the system continues to operate. The DPR allows time-multiplexing of mutually time-exclusive algorithms/steps on a finer scale of the available resources and is characterized by shorter reconfiguration times since FPGA configuration time is directly proportional to the configuration bitstream size. The DPR can also be used for applications such as mitigation and recovery from single-event upsets (SEU) and for real-time dynamic scenario-based adaptive image processing. Furthermore, the OPU also has a “golden image” that enables booting a previous version of a steady on-board processing configuration. The OPU will automatically revert to the “golden image” in case of corruption or unwanted updates or upgrades that have been uploaded from the ground. This is further discussed in chapter 10.

Table 2.2.: Performance f r selected hyperspectral imager modes

Type	Mode A	Mode B	Mode C	Mode D	Mode E
ADCS Mode	Slew	Slew	Slew	Nadir	Nadir
AoI (pixels)	1080×684	1080×684	1080×1194	1080×1194	1936×1216
Binning, B_λ (pixels)	9	18	9	9	1
Spectral range (nm)	$387 - 801$	$387 - 801$	$387 - 801$	$387 - 801$	$220 - 967$
Spectral bands	120	60	120	120	215
Bandpass $\Delta\lambda$ (nm)	3.33	6.67	3.33	3.33	3.33
Frame rate FPS	18	12	12	12	10
Exposure time, τ (ms)	51	79	79	79	96
Scan duration (s)	53.08	53.08	57.00	9.19	1.00
Number of frames	956	637	685	111	10
Scan distance, along-track (km)	40.08	40.08	69.97	69.97	7.60
Swath width (km)[a]	40.08	40.08	69.97	69.97	69.97
Spatial resolution, along-track (m)[a,b]	553	582	618	1101	1231
Spatial resolution, cross-track (m)[a,b]	58.60	58.60	58.60	58.60	58.60
SGSD, along-track (m)	57.6	86.3	124.1	634.4	761.3
$\text{SNR}_{\text{water},[B_\lambda,1]}$ @470 nm[a]	158.1	196.3	197.4	197.4	217.9
Data size, raw (MB)	156.94	52.28	196.29	31.81	4.71
Data size, MOBIP (MB)	69.75	23.24	87.24	14.14	2.09
Onboard processing time (s)[c]	5.8	1.9	7.2	1.2	0.2
OPU-PC transfer time (s)[c]	1924.1	641	390	2406.7	57.7
Downlink time (s)[d]	558.0	185.9	697.9	113.1	16.7
Data size, DROBIP (MB)[e]	11.62	7.75	14.54	2.36	0.22
Onboard processing time (s)[e]	5.6	2.2	7.0	1.1	0.2
OPU-PC transfer time (s)[c]	320.7	213.7	401.1	65.0	6.0
Downlink time (s)[d]	93.0	62.0	116.3	18.8	1.7
Data size, TOBIP (MB)[e]	1.31	0.87	1.64	0.27	0.02
Onboard processing time (s)[e]	8.9	3.0	11.1	1.8	0.29
OPU-PC transfer time (s)[c]	36.1	24.0	45.1	7.3	0.7
Downlink time (s)[d]	10.5	7.0	13.1	2.1	0.2
Data size, COBIP (MB)[e]	0.33	0.22	0.41	0.07	0.01
Onboard processing time (s)[e]	23.6	7.9	29.6	4.8	0.8
OPU-PC transfer time (s)[c]	9.1	6.1	11.4	1.9	0.4
Downlink time (s)[d]	2.6	1.8	3.3	0.6	0.1

a: viewing at nadir.
b: the spatial resolution in one frame, not the final image resolution using e.g. image registration and super-resolution.
c: includes time used for running on memory in the OS and writing data to SD-card at 100 Mbps.
d: total time required for 1 Mbps downlink data rate with S-band radio.
e: estimated based on Table 2.1.

Table 2.3.: Mode A Data Latency for HYPSO-1 on example date 28 May 2022

	MOBIP	(69.75 MB)	DROBIP	(11.62 MB)	TOBIP	(1.31 MB)	COBIP	(0.33 MB)
Sequence	Start time	Duration (s)	Start time	Duration (s)	Start time	Duration (s)	Start time	Duration (s)
Orbit 1								
Image acquisition	10:29:40.0	53.1	10:29:40.0	53.1	10:29:40.0	53.1	10:29:40.0	53.1
Onboard processing	10:30:33.1	5.8	10:30:33.1	5.6	10:30:33.1	1.5	10:30:33.1	23.6
OPU to PC transfer	10:30:38.9	1924.1	10:30:38.7	320.7	10:30:34.6	36.1	10:30:56.7	9.1
Downlink NTNU	-	-	-	-	10:31:10.7	10.5	10:31:05.8	2.6
Downlink KSAT Spain	-	-	10:35:59.4	93.0	-	-	-	-
Cruise	11:02:43.0	434.7	-	-	-	-	-	-
Eclipse	11:09:56.7	2145.0	-	-	-	-	-	-
Orbit 2								
Cruise	11:45:42.0	630.0	-	-	-	-	-	-
Downlink KSAT Svalbard	11:56:12.0	276.0	-	-	-	-	-	-
Downlink NTNU	12:00:48.0	282.0	-	-	-	-	-	-
Total latency (min)		95.85		7.87		1.69		1.47

Table 2.4.: Mode A Data Latency for HYPSO-1 without CAN overhead on example date 28 May 2022

	MOBIP	(69.75 MB)	DROBIP	(11.62 MB)	TOBIP	(1.31 MB)	COBIP	(0.33 MB)
Sequence	Start time	Duration (s)	Start time	Duration (s)	Start time	Duration (s)	Start time	Duration (s)
Image acquisition	10:29:40.0	53.1	10:29:40.0	53.1	10:29:40.0	53.1	10:29:40.0	53.1
Onboard processing	10:30:33.1	5.8	10:30:33.1	5.6	10:30:33.1	1.5	10:30:33.1	23.6
Downlink NTNU	10:30:38.9	316.0	10:30:38.7	93.0	10:30:34.6	10.5	10:30:56.7	2.6
Downlink KSAT Spain	10:35:54.9	242.0	-	-	-	-	-	-
Total latency (min)		8.62		2.53		1.09		1.32

2.2.5 Ground Support

Indicated at the bottom right in Figure 2.3, some of the algorithms should operate on the ground at all times to (a) adjust, fine-tune and prepare data for end-users; (b) assist in in-orbit calibration of the hyperspectral imager, and (c) rigorously test accuracy and reliability in algorithm updates before uploading them to the satellite for on-board image processing. Advanced modules such as image registration, geo-referencing, atmospheric correction, and super-resolution are dedicated for use on the ground because they require access to prompt reference libraries and are computationally expensive. In-orbit upgrades, or at least future missions, may include versions of the algorithms as mentioned earlier only if maturity is demonstrated on the ground. Apart from suitable implementations before launch, alternative prototype algorithms for dimensionality reduction, target detection and classification are also tested on the ground first.

2.2.6 Data Latency in Typical Hypso-1 Operations

Table 2.2 shows HYPSO's remote sensing performance for selected hyperspectral imager modes and corresponding size and latency for data products obtained from MOBIP, PCA-based DROBIP, TOBIP with one two-dimensional map, and COBIP with 16 classes. For each pipeline, the assumptions are based on the chosen spectral channels, the pixel size in bits, reduction factors, and processing speeds stated in Table 2.1, but with extended results for other datacube sizes. The ADCS modes with slew maneuvers assume no attitude control and knowledge errors. The SNR is calculated using the ToA water-leaving radiance based on data from MOBY as described in [23].

Modes A and B provide higher spatial resolution but narrower FOV for a chosen observed area size of approximately 40 km by 40 km, while Modes C and D provide coarser spatial resolution and wider FOV for a chosen target area size of approximately 70 km by 70 km. Modes A, B, C and D use 1080 out of 1936 spectral pixels to cover the relevant spectral range of $400 - 800$ nm. Mode E is dedicated for diagnostics and in-orbit calibration during commissioning phase. "Onboard processing time", "OPU-PC transfer time" and "Downlink time" are the durations needed for image processing for selected pipeline, completing the data transfer between OPU to PC at speeds of up to 290 kbps and completing the data downlink to ground through S-band radio at a bandwidth of 1 Mbps, respectively. It is also assumed that the onboard data is written to the SD-card at 100 Mbps which is included in the onboard processing time.

The results from Table 2.2 are put into the context of a typical mission scenario where HYPSO-1 uses Mode A to observe a 40 km $\times$ 40 km near Lofoten, Norway, then immediately aims to downlink a selected data product to ground stations at NTNU, KSAT Svalbard and KSAT Spain with respective elevation angles assumed to be $5°$, $2°$ and $8°$. Using simulated orbit propagator in Analytical Graphics, Inc., (AGI) Systems Toolkit (STK) with epoch date set to 28 May 2022, results are shown in Tables 2.3 with OPU-PC overhead and 2.4 without the overhead. A dash indicates that the operation is not available or necessary. "Cruise" means that HYPSO-1 is only harvesting solar energy and "Eclipse" means that it is in the Earth's shadow. With overhead in OPU-PC transfer, all except for MOBIP data product can be downloaded in less than one orbit, or specifically less than 10 min. All data products are available in less than 10 min without the overhead.

Regarding the OPU-PC transfer overhead, the current hardware and software architecture in HYPSO-1 is limited by the communication interface between the OPU and the PC due to data transfer over a CAN network with a data rate of about 300 kbps, which negatively impacts the overall latency for higher data volumes as indicated in Table 2.3. In future missions, the physical interface could be replaced with a data bus capable of higher data rates, for example, Ethernet or RS-422, which would involve spending much less time by downlinking directly from the OPU, and better latency can be potentially achieved as shown in Table 2.4.

2.3 Conclusions

Following the advancements in miniaturization, image processing algorithms and sensor technology, the mission and system design of HYPSO-1 shows that pushbroom hyperspectral imaging combined with FPGA-based on-board image processing on a nano-satellite, can enable ocean color data products with high spatial and spectral resolution and low data latency to meet the user needs for operational coastal environment monitoring. The imager design, HYPSO-1's remote sensing approach, and on-board software grant flexible trade-offs to be made between spatial image resolution, spectral resolution, and SNR. The chosen FPGA-based CCSDS123v1 lossless compression, dimensionality reduction, target detection, and classification algorithms may reduce the data size significantly without losing crucial information. Contrary to using rigorous data processing and analysis on the ground, the smaller data products can be made available shortly after observation. This enables quick download of tailored data products that may satisfy the immediate needs of end-users. As such, it could allow better mitigation for potential damage from Harmful Algal Blooms when early detection and warning are needed. Based on lessons learned from the HYPSO-1 operations, the goal is to iterate and enhance the design of the hyperspectral imager, attitude determination and control system, satellite communications architecture, and the on-board image processing algorithms for future missions. After launch, the HYPSO-1 mission aims to determine the efficacy in quickly providing high-resolution hyperspectral data from small satellites for ocean color applications.

The figures of merit presented in this paper such as optics size, spatial resolution, spectral resolution, swath width, SNR, Sequential Ground Sampling Distance, data latency as well as spacecraft angular

velocity and attitude accuracy, can be used for system trade-off studies in the design of a remote sensing mission.

Acknowledgments

The authors would like to thank Kanna Rajan at Faculty of Engineering, University of Porto, for his substantial contribution to the HYPSO-1 mission design and motivation towards establishing a small-satellite constellation for oceanography. Authors also thank Liane S. Guild, Stephen Dunagan and Chad Frost at NASA Ames Research Center, Raphael M. Kudela at the Ocean Sciences Department, University of California Santa Cruz, Richard P. Stumpf at National Oceanic and Atmospheric Administration (NOAA), and Ajit Subramaniam at the Lamont-Doherty Observatory, Columbia University, for their guidance and valuable discussions on remote sensing and mission development. Authors would also like to thank Geir Johnsen at Department of Biology, NTNU, for his review on requirements in ocean color remote sensing, Torbjørn Skauli at Norwegian Defense Research Establishment (FFI) for his review on spectroscopy, Fernando Aguado Agelet at the Department of Telecommunications Engineering, University of Vigo, and Cecilia Haskins at Department of Mechanical Engineering, NTNU, for their input on systems engineering in the project, Annette Stahl and Dennis Langer at Department of Engineering Cybernetics, NTNU, for their strong inputs on image processing, Egil Eide and Gara Quintana Diaz at Department of Electronic Systems, NTNU, for their help on radio communications, and Harald Martens, João Fortuna and Petter Rossvoll at IdleTechs for their recommendations on dimensionality reduction of hyperspectral data. Finally, authors are grateful for the participation of NanoAvionics and their continuous feedback throughout the project.

Chapter 3

Oceanographic Observation with Multiple Platforms

Collaboration is being open to each other's ideas and benefiting from each other's perspectives in an open way.

Alan Menken

The extensive data volumes associated with hyperspectral images and the computational demands required to interpret them make acquiring, distributing, and utilizing them more complicated. The NTNU is building an ocean observation system, where the HYPSO-1 hyperspectral imaging satellite is at the apex of the observational pyramid introduced in chapter 1, to monitor ocean color in coastal waters. The images collected by HYPSO-1 are intended to inform an ensemble of robotic agents that locally monitor ocean conditions. Several of the agents will be UAVs that also carry hyperspectral imagers, but at a lowered altitude below the clouds. The hyperspectral images will be processed on-board the different agents and central locations. This chapter provides a draft of the architecture, development, current status, and future opportunities of hyperspectral image processing pipelines on these platforms.

This chapter describes the Hyperspectral Image Processing Pipelines (HIPPs) developed for the HYPSO-1 satellite and UAVs carrying hyperspectral imagers. Furthermore, additional capabilities developed on the ground to facilitate their operation are also described. Each platform has its purpose and constraints that guide the design and development plan of each HIPP, and they come together as part of the MASSIVE project.

The project is intended to consist of multiple small satellites, the first of which is HYPSO-1 described in chapter 2 [116], combined with an ensemble of robotic agents, such as UAVs and USVs. These robotic agents will be used as observation platforms to validate ocean color observations and relate ocean color to local conditions [24, 117] (Figure 3.1). The acquired data will be processed

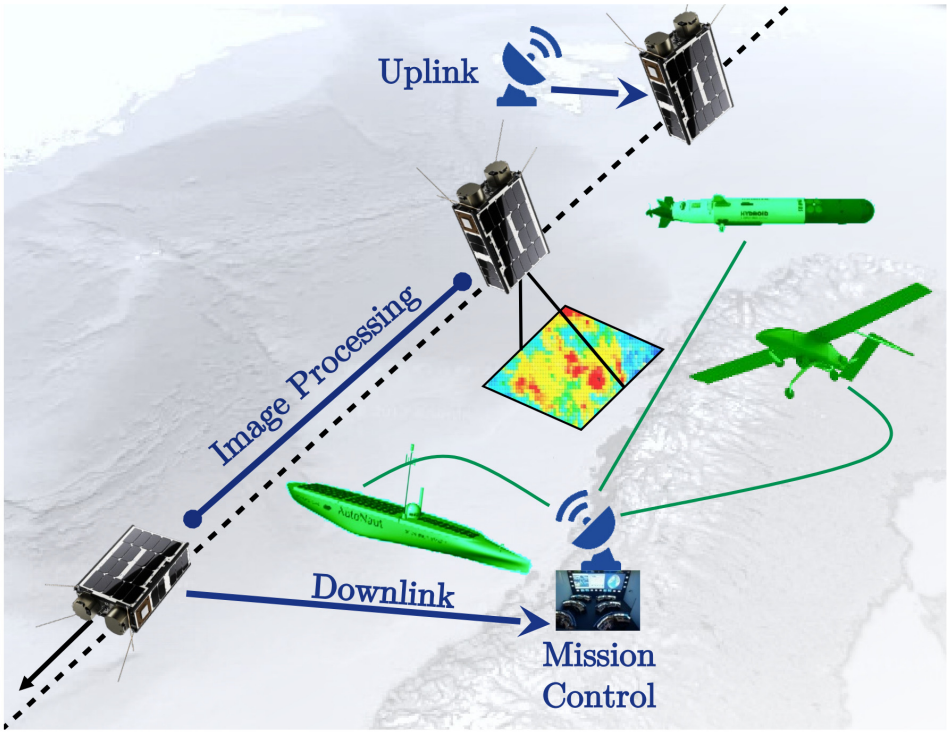


Figure 3.1.: The hyperspectral observations of the HYPSO-1 satellite will inform an ensemble of *in situ* autonomous agents. Illustration by Joseph L. Garrett

on the ground at the mission control center, before being dispersed to end-users. This chapter is intended to provide further details on how hyperspectral imaging is planned to be integrated into a multi-agent system and will describe the image processing tools built to facilitate its use. Particular attention will be placed on the interfaces between the different HIPPs and how those interactions motivated design choices.

3.1 Purpose of Each Pipeline

The HYPSO-1 satellite will as a small satellite in low Earth orbit have a revisit time of approximately 90 minutes. This frequency should enable the satellite to point towards algal bloom events and image their dynamic evolution up to three times a day.

A primary goal of image processing on-board the satellite is to reduce the size of the data, while retaining as much helpful information as possible, as briefly discussed in Chap 2, and will be the topic of the subsequent chapters. The expected size of the raw data is a few hundred MB, and this will take several passes to be downlinked in its raw form with the communication links available. However, the mission concept aims to reduce its size by on-board processing.

Table 3.1.: Modules considered for the pipelines

Module	Purpose
Acquisition	To collect spectrograms
Compression	To reduce size
Sensor cor.	To eliminate systematic errors
Image registration	To place pixels on a grid
Georeferencing	To position the data on Earth
Super-resolution	To emphasize spatial information
Atmospheric cor.	To mitigate atmospheric effects
Dim. reduction	To extract crucial information
Target detection	To locate target spectra
Classification	To partition image using spectra

Furthermore, once the data is downlinked, there are three potential ways of utilizing the data

1. The downlinked data can be processed further on the ground and transmitted to the autonomous agents
2. the downlinked data can be processed on the ground for storage in some data repository
3. the downlinked data can be transmitted to the autonomous agents without further processing.

The UAVs are planned to both supplements the satellite observations and image ocean phenomena independently. As an example, during conditions with high cloud coverage, when HYPSO-1 is unable to observe, the UAVs will be used to acquire images beneath the clouds. As these hyperspectral imagers are passive sensors and dependent on the sun or reflections from the sun as a light source, this can pose a challenge in and of itself. In better weather conditions, the UAV based platform should be able to acquire images at better spatial resolution and better signal-to-noise ratio (SNR) than what is possible with HYPSO-1, albeit be able to cover less area. This data from the UAVs is intended to be used to calibrate the satellite's hyperspectral camera with simultaneous nadir overpass [32], and to sample regions at higher temporal resolution for further data assimilation and integration. The primary goal of the HIPPs on UAVs is to reduce the size of information so that it can be stored on-board with an acceptable information loss. The data loss as a result of compression can be minimal as the UAV, and the collected data can be physically transferred to the operator or have better bandwidth on the communication links, unlike data on the satellite. The navigational system of the UAVs could also use the hyperspectral images collected by the drone to perform adaptive sampling such as circling the perimeter of ocean events, such as an algae bloom [24, 118]. Although the UAVs may be carrying different models of cameras than the satellite, many of the image processing modules are intended to be and will be reused.

The purpose of the analysis capabilities developed for use on the ground is to aid the UAVs and satellites in their mission. For example, for the data to be interpretable, it is necessary to know what wavelength each column in a spectrogram corresponds to. Before a flight, the radiometric, spectral, and spatial properties of the response of the camera are calibrated and parameters are

determined to convert data into the desired format [81]. Applying this calibration, to the data on-board the agents as it is collected can standardize it and eliminate undesired systematic effects before further on-board processing. The ground pipeline is used to parametrize the hyperspectral camera characteristics. This information can, in turn, be applied on-board the remote platforms. In addition, the ground HIPPs will also be used to coordinate the satellites and other agents. For example, the satellite data can be used to determine where the *in-situ* agents should go.

Constraints These pipelines are being developed to support hyperspectral imagers similar to the one described in [5, 69]. Different variants of the camera are being produced for use on either the satellite or UAV, but their core design and operation remain similar.

The computational hardware differs between the platforms. As mentioned in chapter 2, the satellite uses a system-on-a-chip with a customized carrier board, which can utilize an on-chip Field Programmable Gate Array (FPGA) for low-power accelerated computations. A diagram showing the relevant components are given in Figure 3.2. On the other hand, the two UAV imaging payloads lack FPGA acceleration, but do include Graphics Processing Units (GPUs) for more effective computing. One version of the UAV payload is oriented towards tagging the acquired spectrograms with time and pose, whereas the other is designed to enable users to operate the system. The ground analysis software is designed to be run on a laptop, so that it can be used while operating UAVs, or a desktop, to be run at the Mission Control Center.

Because of differing choices for the computational hardware, the algorithms utilizing either FPGA or GPU acceleration cannot be run on both the satellite segment and the UAV segment. Unfortunately, this imposes a high cost for developing in parallel for these systems. Therefore, algorithms will be tested as part of a pipeline on the ground, where more computational resources are available, before being developed for the other systems. For both the satellite and the UAVs, there is also a time constraint on the computation. Either the data must be processed as they are collected, or the agent must alternate between imaging and processing.

3.2 Design and Development

The following two sections describe how the spacecraft and UAV are designed. It focuses on what kind of electronics hardware is used on the different platforms. The chosen electronics hardware is intended to be low in power usage while also effectively processing the acquired hyperspectral imagery.

3.2.1 Satellite

The HYPSO-1 satellite will image a region of the ocean for about one minute and process the acquired data for at most three minutes more as discussed in Chap 2 [116]. From its conception, the HYPSO-1 satellite on-board processing pipeline was planned to incorporate the image processing modules listed in Table 3.1. Several image processing pipelines, each containing a subset of the modules, are in development, with each designed to produce a particular data product.

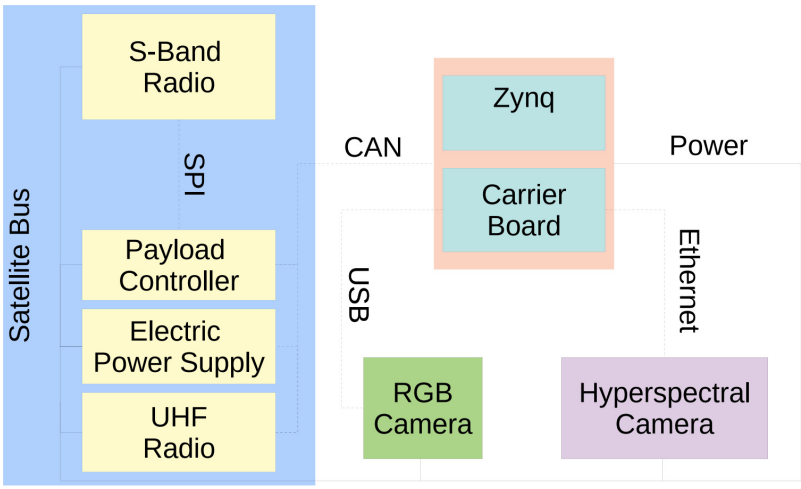


Figure 3.2.: The camera payload on the satellite included the hyperspectral imager, an on-board processing unit, and an RGB camera. The blue highlighted regions show the spacecraft subsystems that the payload interacts with. Power is supplied over the solid lines, while data is transmitted over the dashed lines. Note that only data acquisition, processing, and downlinking subsystems are pictured. Illustration by Joseph L. Garrett

The HSI camera, an RGB camera, and the On-board Processing Unit (OPU) together form the payload of the satellite. The OPU consists of a Zynq 7030 System-on-Chip (Xilinx) together with a custom carrier board, as illustrated in Figure 3.2. The HIPP runs on an embedded Linux Operating System (OS), and the modules are partitioned between Central Processing Unit (CPU) and FPGA, and therefore written in either C or VHDL, respectivly The interfaces between the FPGA and the OS are implemented using Linux kernel modules. A Linux-based OS is used (not a real-time OS or a bare-metal solution), due to proprietary software needed to interface with the COTS camera module used in the hyperspectral imager. The cubeDMA interface is used to facilitate flexible communication of the hyperspectral data structures between memory and FPGA [77].

The compression and dimensionality reduction modules are prioritized because they directly reduce the size of the data and can be utilized in more configurations of the HIPP [80]. The simplest of the configurations for the satellite, which is considered the Minimum Viable Product (MVP) or minimal pipeline, consists of image acquisition, compression, and downlinking, and is depicted in Figure 3.3. The compression module follows the CCSDS-123 standard and is implemented in on FPGA [78, 119]. An alternate version of the compression code which runs on the CPU can also be run as backup, in case some aspect of the FPGA fails to operate as expected in space. However, the CPU version takes much longer to process, dependent on the acquired cube dimensions, so the FPGA implementation is preferred for operational reasons. Sensor corrections, such as compensation for smile and keystone, are prioritized next. The following configuration builds on the first, but also adds a smile and keystone sensor corrections [81] and dimensionality reduction [84]

to the pipeline. The smile and keystone corrections are included because dimensionality reduction removes information that is necessary to correct for smile and keystone, which preclude applying the corrections after the dimensionality reduction data are downlinked. Or conversely, with the smile and keystone effects the dimensionality reduction algorithm will not be able to generalize well across the perceived spectral bands of the acquired cube. It is possible to change the number of components that are selected from dimensionality reduction to meet the memory requirements of a specific downlink rate. Nominal operations include downlinking the 20 bands resulting from dimensionality reduction, most relevant for reconstructing the original image cube. This dimensionality reduction will reduce the size of the data package to 20%. The pipeline that adds both the smile-and-keystone correction and the dimensionality reduction is known as the baseline, given in Figure 3.3. Other modules that have been prototyped on the ground for incorporation into the satellite HIPP include super-resolution and target detection [120–122].

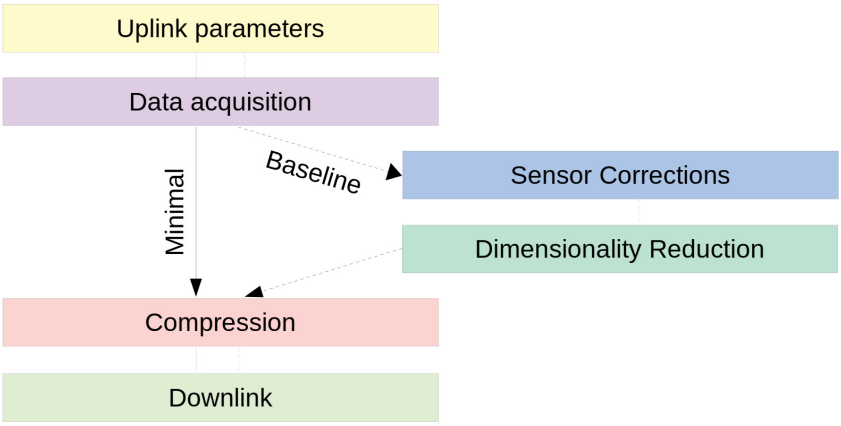


Figure 3.3.: The minimal on-board image processing pipeline provides the basic structure that all more sophisticated pipelines will build on. The baseline on-board image processing pipeline adds dimensionality reduction to increase the speed at which the data can be downlinked. Illustration by Joseph L. Garrett

3.2.2 Unmanned Aerial Vehicle

Several different payloads have been developed to acquire and process images. This development has provided a simple way to mount the hyperspectral imager on various drones. The original payload, called the Lunde, was developed for mounting the camera on UAVs and acquiring images, and is described in [123]. Because different camera users had different demands, two new versions of the payload have been designed.

The first payload consists of a processor (Odroid XU-4, Hardkernel) together with a voltage converter, a 1 TB solid-state drive (PNY), and an ethernet connection to the drone (Figs. 3.4 and 3.5). This payload is designed to complement the accessibility of the do-it-yourself camera [69] because it is composed of relatively cheap commercially available components. The Odroid controls

the camera through the DUNE Unified Navigational Environment [124], which in turn controls the uEye (IDS) camera, driver. Imaging automatically begins when the Odroid is powered to make the payload easy to use. The acquired data can then be accessed directly through the hard drive so that there is no need for the operator to modify the payload. However, for more flexibility, the camera can also be controlled remotely through the DUNE software.

The second payload is designed for more precise scientific measurements. It is based on the Jetson TX2 processor (Nvidia) and follows the same basic configuration of the first payload, but adds an IMU (Analog Devices Inc. ADIS 16490 BMLZ) and a GNSS system (module: u-blox Neo-M8T, antenna: Harxon HX-CH3602A). A reconfigurable timing board (SenTiBoard) is used to synchronize and align the timing information of the camera with the pose [125]. Knowing the pose of the UAV is critical for determining the georeferencing the hyperspectral data.

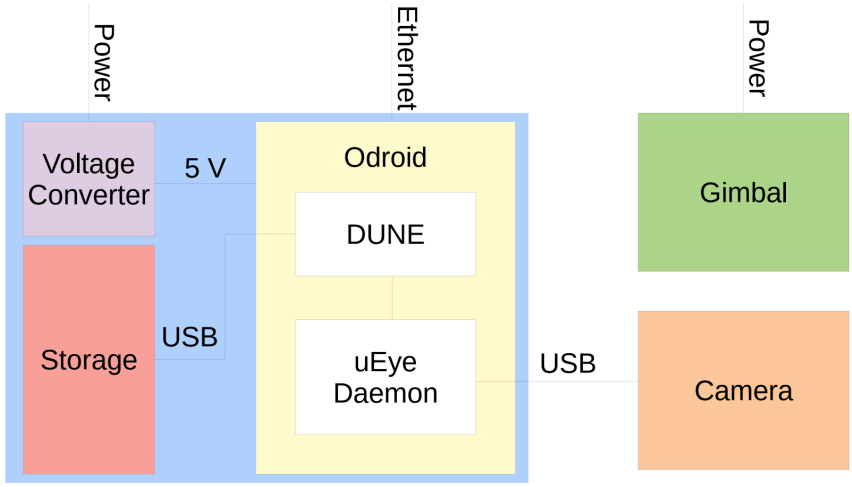


Figure 3.4.: The hyperspectral camera UAVpayload is based on an Odroid which runs DUNE which in turn controls the uEye daemon to operate the camera. Illustration by Joseph L. Garrett

To reduce size and weight, some camera operators have wanted to fly the camera without an additional computing payload. To facilitate this, a python script based on the pypyueye library is used to control the camera [126]. While this lacks the precise timing information and remote control through DUNE, it requires less additional software to be installed. A secondary benefit that we have encountered is that this implementation is straightforward to integrate with modules developed from other organizations into the pipeline, due to the widespread adoption of Python. For example, the smile and keystone correction developed and tested for the ground pipeline can be run directly without alteration in this version of the pipeline.

Ground HIPPs are used to develop and test modules to be run on the UAV. For example, in the classification task, the spectra of relevant classes are determined on the ground are evaluated on example data sets before being run in the field. Off-line analysis of the Hyperspectral Imager of

the Coastal Ocean [127] suggests that the classification technique described in [84] is sufficient to distinguish between different constituents such as chlorophyll and pollution. Experiments are currently being prepared to test the performance of classification while the drone is flying.

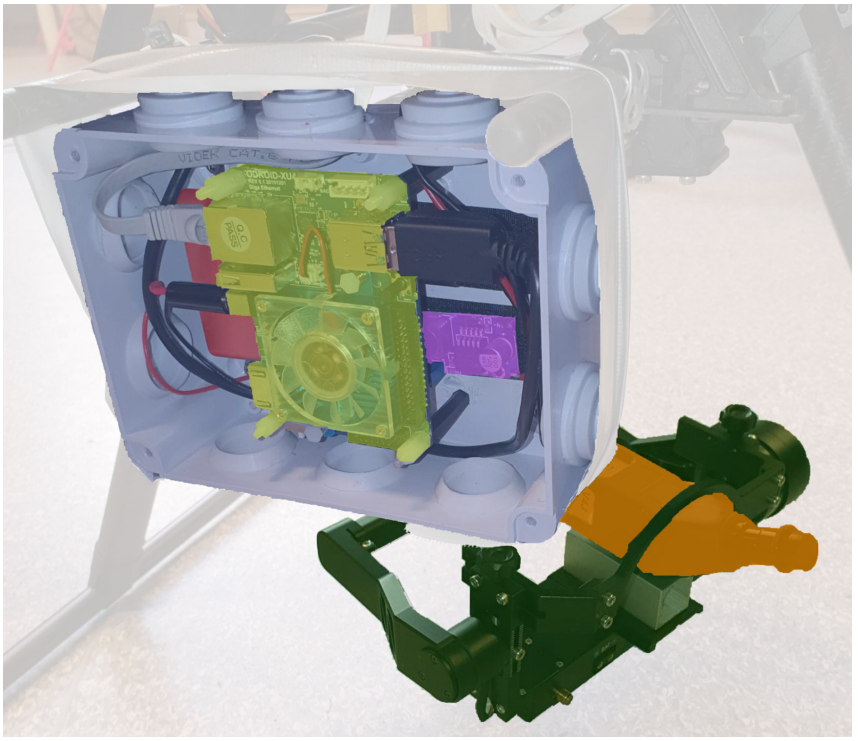


Figure 3.5.: The UAVhyperspectral payload mounted on a multiroter. Note that the the components are colored in order to correspond to Figure 3.4. Photo by Joseph L. Garrett

3.3 Conclusions

This chapter presented the purposes, constraints, design choices, and current development progress of several hyperspectral imaging systems and processing pipelines under development at NTNU. The systems that are described are intended to be united in the MASSIVE ocean observation project, briefly detailed in chapter 1. However, the initial flight version of the HIPP that will be on the HYPSO-1 when it launches has been finalized. Changes to that pipeline must either rely on the in-flight update capacity of the satellite or wait to be implemented on the next satellite in the constellation. The software architecture intended to allow in-flight updates is described in chapter 10. Because it is simpler to collect data from UAVs than from satellites, the development of their HIPPs has been somewhat slower, and more attention has been focused on ease of use and tailoring the payloads to different use-cases. For example, the UAV payloads can be augmented with an RGB camera to investigate joint hyperspectral-multispectral imaging strategies, both for

their utility and to study how they might be incorporated on the satellite [128]. Still, as the UAVs become more autonomous and fly longer missions, the computational capacity of their HIPPs is expected to become more relevant during operations.

Chapter 4

yperspe tra Remote Sensing and Anal sis of the Amazon River Plume

Tell me and I forget.
Teach me and I remember.
Involve me and I learn.

Benjamin Franklin

This chapter shows the ongoing processing of hyperspectral radiometer data from two cruises following the Amazon River plume. Ocean color provides information for many applications such as biomass estimation, water quality monitoring, characterization of biogeochemical cycles. The direct relevance to Norwegian waters may not be obvious, but the Amazon River Plume provides an excellent source to observe complex waters where phytoplankton, various sediments, and CDOM are combined. Albeit with other types of species, this type of complexity in the water column is expected to be found near the Norwegian Coast and fjords. Phytoplankton cells can be characterized by how they absorb, scatter, and fluoresce light; it is attractive to retrieve information by their bio-optical properties. A specific spectral resolution and appropriate wavelengths must be used from bio-optical measurements to infer the sought-after information. There is a need to account for other optically active substances such as CDOM and suspended non-algal particles. Here, the bio-optical properties of the amazon river plume are investigated using hyperspectral remote sensing measurements of downwelling irradiance and water-leaving radiance and compared with results from HPLC and SP.

4.1 Introduction

The Amazon River is the largest in the world and discharges more than 200,000 $m^3 s^{-1}$ into the Western Tropical North Atlantic Ocean. At its peak, the low salinity plume extends more than 2000 km from the mouth and covers more than 1 million square kilometers.

The Amazon and the Amazon River represent a crucial area in the global scope for biodiversity and an abundance of natural resources [129]; It contributes to vital parts of human life, such as water for agriculture, transportation, and food. Furthermore, the Amazon River provides a vital ecosystem for countless species. It has been reported that the water cycle balance in the Amazon rain forest and its rivers are threatened by climate change. The effects of climate change are reported as a risk factor that can cause significant loss in this unique area for future generations. This role as a vital ecosystem and the vulnerability of the Amazon River is why it is essential to understand and monitor it. With more information, it is possible to improve decision-making for the area.

High CDOM in the plume can make it challenging to accurately estimate phytoplankton biomass, total suspended matter, and other parameters of interest via multispectral remote sensing of [Chl a] [1, 7, 130, 131]. In the campaigns presented here, hyperspectral remote sensing measurements of both downwelling irradiance and water-leaving radiance using the SBA [132] were made. These optical measurements were taken as near the sea surface as possible. HPLC and SP data were also collected during the campaign. This data was collected at the same stations where the SBA radiometer was deployed. The collected measurements were then used to investigate the details in ocean color spectra presented by a higher spectral resolution than previously obtained for the waters in question.

While the SBA radiometer approach is not able to provide the exact spatial and temporal resolution of air- and space-borne sensors it does mitigate some of the challenges of atmospheric compensation [9, 132] by direct measurements of the downwelling irradiance. The SBA radiometer, while an attractive approach for monitoring in waters of particular interest, can also provide valuable vicarious calibration data for other remote sensing systems [32], similar to the Marine Optical BuoY (MOBY) and buoy for the acquisition of a long-term optical time series (BOUSSOLE)

The radiance measurements used here are of the plume from the Amazon river and surrounding oceanic waters during a field survey in the summer of 2019 and 2021. Furthermore, the SBA measurements were quality controlled using a tilt filter to remove the data points that logged a tilt angle that exceeded 5 degrees from the nadir of the sensor. This paper also proposes an outlier filter that removes artifacts and other measurements. The proposed filter removes the entire SBA measurement when either the downwelling irradiance or water-leaving radiance is considered an outlier in terms of quantile thresholds set by the user. In general, the SBA system acquired over 4000 measurements at each station. The proposed rigorous quality control accepted about 1% of these measurements for use in subsequent analysis. This data formed the basis for exploring if the high spectral resolution from hyperspectral remote sensing can improve our understanding of this type's plume constituents and water bodies.

An implementation of the shade-correction reported in [133] forms the basis for the SBA processing. only the signals that passed the criteria as mentioned earlier were shadow corrected. The shadow correction code is made available here [134]. It is necessary to compensate for the shading effects experienced by the SBA radiometer. This correction is needed as the shading effect is expected to

cause significant errors for waters with many constituents that are by other means characterized by high absorption [133].

The Quasi-Analytical Algorithm (QAA) [7, 135] (with version number), and the OC4 band-ratio algorithm for SeaWiFS [16] derived bio-optical parameters using the resulting spectra after quality control and shadow correction. Atmospheric correction to compensate for the optical signal propagation is not used as the SBA radiometer measures the water-leaving radiance and the solar irradiance at the sea surface. The sought-after signal does not travel through any medium with effects that are compensated for using atmospheric correction before it reaches the sensor. Comparison with results derived from the SBA spectra and in-situ measurements encourage the use of hyperspectral measurements in optically complex waters that are characterized by high concentrations of CDOM, such as the Amazon River plume [136]. The results presented here further advocate advancing ocean color satellites' instrumentation to support a higher spectral resolution.

4.2 Background

Satellites collect remote sensing data of terrestrial, ocean, coastal, and lake ecosystems. With recent advances in sensor technology, it is possible to equip the future satellites and satellites under development with more advanced remote sensing instruments for ocean color [5, 60, 116].

One type of remote sensing instrument gaining popularity for this payload miniaturization is hyperspectral imagers. With hyperspectral payloads on board ocean color, remote sensing satellites, the data collected will be able to infer more accurately the IOPs of more complex ocean ecosystems. That is, we will be more capable of understanding the extensive dynamic range of optically significant constituents in more complex waters, waters that are typically important for human life, and policymaking [2].

In this chapter, we collect pristine hyperspectral data samples using an SBA radiometer and compare the results from this instrument with other methods for deriving IOP's. This investigation is an essential preliminary step to retrieve valuable information from hyperspectral data of complex waters from remote sensing ocean color satellites.

4.2.1 Field Measurements

The radiometer deploying the Skylight-Blocked Approach (SBA) approach presented in [132] collected measurements as part of the EN640 cruise onboard the RV ENDEAVOR between June and July in 2019. There were 21 stations in 2019 and 31 stations in 2021 where water samples for SP and SBA measurements were acquired simultaneously. In 2019 this took place over 20 days and constituted 15 of the stations, while in 2021, this took place over TBC days and constituted of TBC stations. The locations of the different stations are given in Figure 4.1 and Figure 4.2. The cruise followed the Amazon river plume based on measurements of salinity. The observed values and gradients ranged from 15.8 to 34.7 g/kg during the cruise. At each station, water samples

were collected Conductivity for salinity, Temperature, and Depth for pressure of seawater (CTD) measurements. This was done by using Niskin bottles on a CTD rosette. Between 1050 and 3200 mL of water at each station was filtered onto a 25mm GF/F filter for particulate absorption and phytoplankton pigment concentrations measurements using HPLC. In addition, approximately 50 mL taken from the water samples was passed through a 0.2-micrometer filter. The filtrate was used to measure absorption due to dissolved material using a SP.

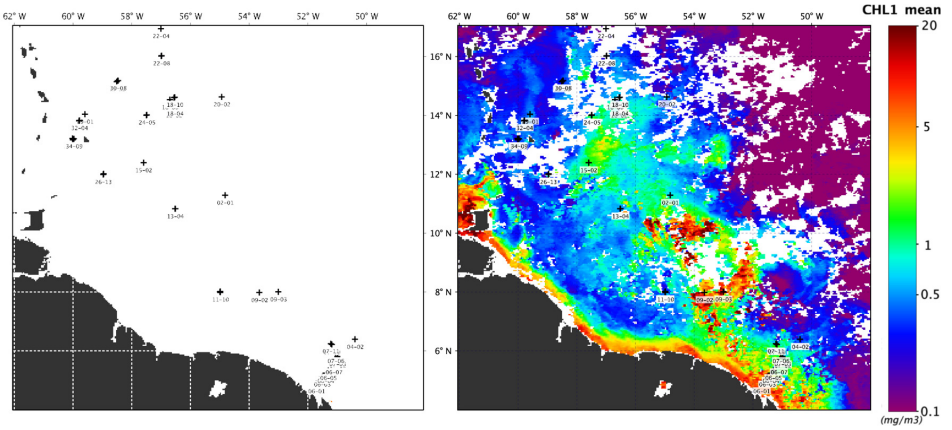


Figure 4.1.: Satellite image of measurement station locations marked with black crosses for the 2019 Cruise.

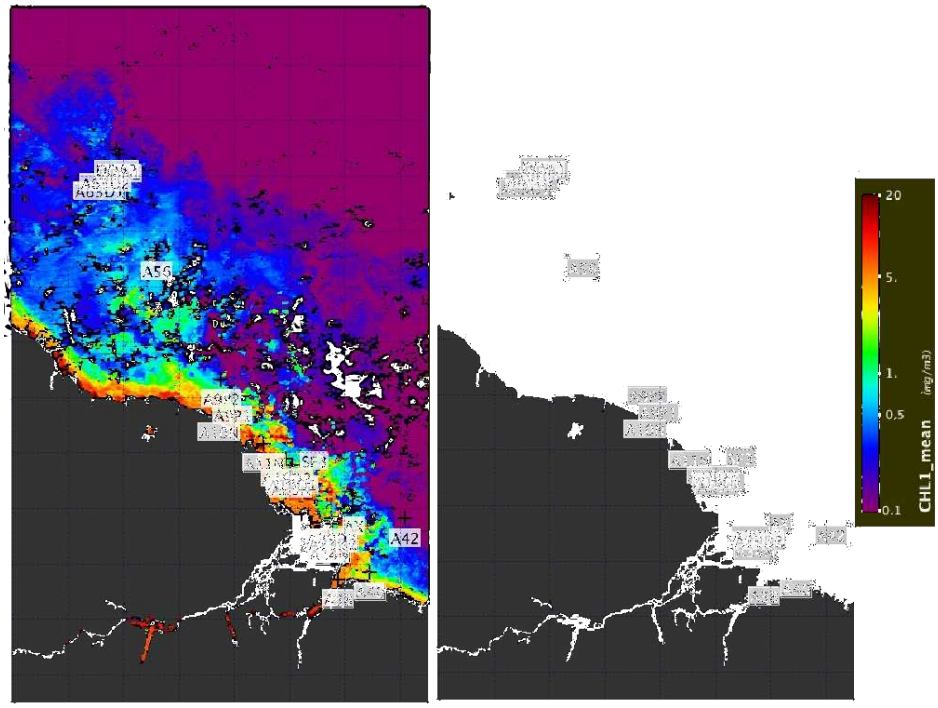


Figure 4.2.: Satellite image of measurement station locations marked with black crosses for the 2021 Cruise.

Hyperspectral Remote Sensing Measurements

The measurements of downwelling irradiance E_s and water-leaving radiance L_w were carried out using a Sky Blocked Approach SBA [132]. This particular realization of the SBA system uses a HyperPro II profiling system from Satlantic, Inc. The resulting system is then able to acquire hyperspectral remote sensing measurements of E_s and L_w in the range of 350 nm to 800 nm with a spectral resolution of 1.5 nm [132]. Figure 4.3 shows a picture of the radiometer deployed in the Amazon River plume.

The SBA radiometer in Figure 4.3 receives light emerging from the water and sky in a nadir direction. During ideal deployment of the SBA, the base of the cone is beneath the water surface, while the fore optics of the radiometer is in the air. The goal of the quality-control filter was to remove measurements that did not fit this description; beyond that which the tilt threshold was able to remove. As the cone blocks reflected sea surface light, only the radiance emerging beneath the water surface is measured. The radiometer position for downwelling irradiance accompanied the radiance sensor on a balanced float, such that recordings of L_w and E_s were taken at the exact location and time. As the measurements are practically in the same spot, mainly measuring the solar radiance flux, there is a reduced need to compensate for different contaminating effects.

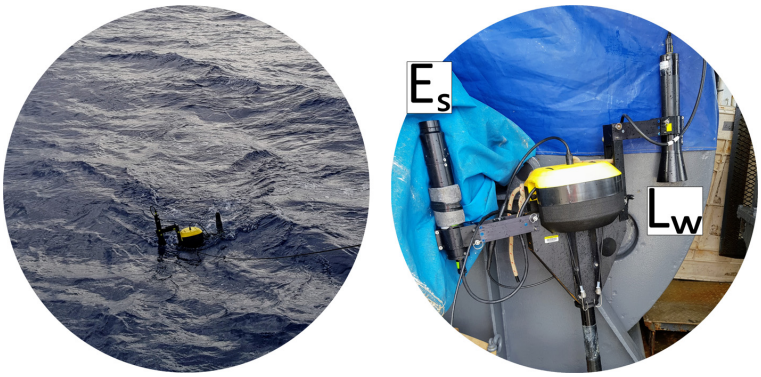


Figure 4.3.: The SBA radiometer with labels for where the downwelling irradiance E_s and water-leaving radiance L_w is measured.

Other Measurements

The campaign also collected in-situ measurements of particulate absorption colored dissolved organic matter and diagnostics phytoplankton pigments. The collected measurements and inferred parameters of interest are compared with those derived from (pristine) hyperspectral measurements.

Absorption properties found in the collected water samples were measured directly using the Shimadzu UV-2401PC SP. This instrument is depicted on the right side of Figure 4.4. This single monochromator system boasts having a low stray light and wide dynamic range and captures the finer details of the samples tested [137]. More specifically, the stray light effect is reported to be less than 0.015% at 220nm and 340nm, and the SP can record results at 0.1nm resolution with a spectral accuracy of ±0.3nm. In the configuration used during the in-situ recording, a spectral resolution of 0.5nm is used. It works best when there is a solid signal to measure.

Additionally, the collected samples from the 2019 cruise were analyzed using HPLC [29, 136], which is a chromatographic method to more accurately infer the properties that reside in the collected water samples. This method separates, identifies, and quantifies each filtrated water sample's particulate absorption and phytoplankton pigment concentrations. The HPLC results for the 2021 cruise are still pending.

The method utilizes pressurized liquid, a solvent containing the sample mixture, through a column filled with a solid adsorbent material. Each component or pigment elutes, that is, is removed as an adsorbed material from an adsorbent using a solvent, at a different time due to the differences in polarity of the different pigments [136]. This difference in elution will result in different flow rates for each of the various pigments. The recorded flow rate associated with each pigment is a latent variable to separate them as they flow out of the column.

Figure 4.4.: Tools and methods used during sampling. The image on the left depicts a rosette station with Niskin bottles for CTD measurements, while the center image shows the collection of water from the rosette station. the rightmost image shows the spectrometer which was used to analyze the absorption properties of the collected water samples in-situ. Center photo shows me collecting data during the 2019 Cruise.

4.3 Methods

4.3.1 Data Processing

Ocean color measurements from satellites are today mostly multispectral. The ocean color measurements are converted to IOPs by Semi-Analytical Algorithms (SAAs), and the operational satellites do currently provide estimates of IOPs across the entire world daily [2, 138]. In addition, the collected ocean color data also derive other parameters of interest, such as estimates of chlorophyll concentrations.

Most SAAs are constructed similarly. However, few, if any, are appropriate for all water masses, nor all seasons [1, 7, 131, 138].

As all SAA models are approximations, some are more applicable than others. There have been attempts at making a unified model, namely the Generalized Inherent Optical Properties (GIOP) [138]. The IOP model is a combination of different SAAs that an operator at runtime can modify, whilst the QAA is one of the more generally accepted aforementioned SAAs [135, 138, 139], and is a significant contribution to the GIOP model. How the data processing and pre-processing of spectral data is conducted will affect the interpretation of every intermediate step and the final results. For multivariate data processing, e.g., processing of hyperspectral data, this should be considered carefully as the interpretation of the final results may have a considerable impact on the analysis and beyond [2, 15, 140].

Data Pre-Procsessing

This section provides details on how the collected data from the SBA radiometer is pre-processed before furthere analysis. The analyzed spectra need to be satisfying to ensure that the QAA provide the best possible output. Suppose the analyzed spectra do not represent the type of spectra. In that

case, the QAA expects the output will not be able to infer the IOPs and following parameters of interest well [7]. The pre-processing procedures proposed here for the data collected by the SBA radiometer consist of the following four steps.

1. Removal of missing values and bad sensor readouts.
2. Removal of spectra associated with an absolute viewing angle exceeding 5.0° from nadir.
3. Removal of spectra with outlier characteristics determined by qunatiles.
4. Removal of shading effects from instrument [133]

In the initial step, the faulty sensor readouts are easily defined by non-physical values for E_s and L_w. An angle sensor aids the secondary pre-processing procedure to determine the nadir angle of the sensor during data acquisition. The fidelity of the angle sensor is deemed sufficient, and the registered angle is used as a threshold directly and without any modification before further processing.

In the pre-processing scheme, the outliers are determined by manually setting desirable quantiles, and here the threshold was arbitrarily set at 10 and 90 percent. This pre-processing step resulted in spectra that appeared to be sensible and did not seem to include any contaminated by the expected effects, such as ocean whitecaps.

The quantiles are being computed using an *exact* sorting-based algorithm [141]. An illustration of the third step, the outlier removal, is given in Figure 4.5. The median for each wavelength is computed for the spectra that pass through the quantile filtering, and the resulting median spectra are shadow corrected. Further analysis is performed on the resulting spectra.

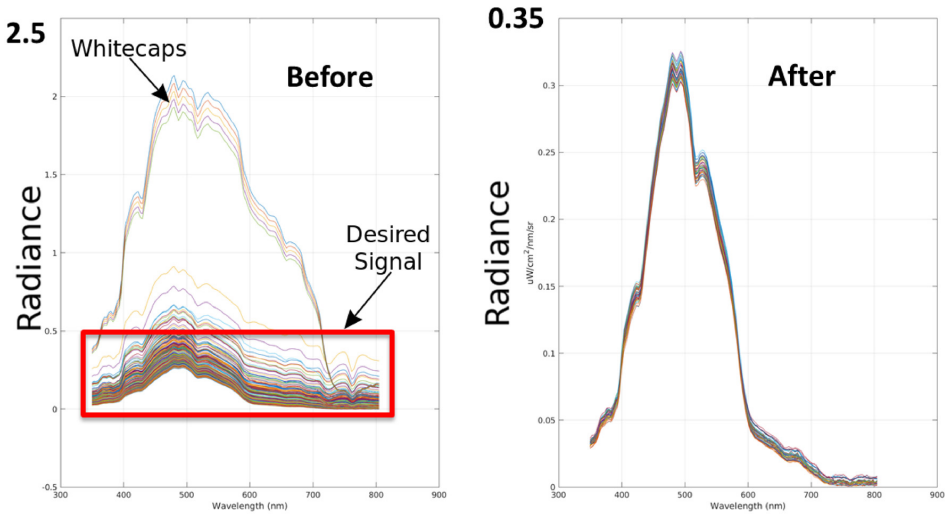


Figure 4.5.: How the removal of spectra with outlier characteristics using quantiles affects the data.

As demonstrated in [133], one of the challenges when using the SBA radiometer is to account for the shading effects. Determining the solar zenith angle in combination with the geo-location of the acquired measurements can aid in compensating for the shading effects. Here, the shadow correction is performed as found in [133].

4.3.2 The Quasi-Analytical Algorithm (QAA)

Originally the QAA was developed for open ocean and coastal waters, as a multispectral approach to retrieve total absorption a_{tot}, absorption due to phytoplankton pigments a_{ph} and absorption due to CDOM, such as, detritus and gelbstoff absorption a_{dg}, as well as backscattering b_{bp} [7,135]. With the QAA approach, these sought after IOPs are analytically derived from values of remote-sensing reflectance-based radiative transfer models. The absorption due to phytoplankton pigments and detritus and gelbstoff is derived by spectrally decomposing the total derived absorption. Thus, the QAA approach can, and has, been used on hyperspectral data [7,132,133,135]. As there are limited empirical relationships used and only used for less critical steps, the QAA can be applied for all types of oceanic observations [7].

Details of the implementation of QAA can be found in [7,135], but for completeness relevant parts are restated here. The QAA approach consists of 11 steps, given in Table 4.1 taken from [135], updated to version 5 [142] dubbed QAAv5. In the implementation used here, 555 nm is used as the reference wavelength λ_0. In Table 4.1 mathematical notation of the proprties column is explained in [7]. Here, $r_{rs}(\lambda)$ is the remote sensing reflectance just below the surface. $u(\lambda)$ is a polynomial approximation of the ratio of backscattering over total absorption and backscattering. $a_{tot}(\lambda)$ is the total absorption. $b_{bp}(\lambda)$ is the backscattering coefficient of particles. Y is the exponent for particle backscattering coefficient. a_ph Absorption coefficient of phytoplankton pigments. a_{dg} Sum of absorption coefficients of non-algal particles plus yellow substances. ζ is the spectral ratio of $a_{ph}(411)/a_{ph}(443)$. ξ is the spectral ratio of $a_{dg}(411)/a_{dg}(443)$.

4.4 Results and Discussion

4.4.1 Data Presentation

In this section, the collected data is presented. The collected data from the HPLC with match-ups with the SBA radiometer is given in table 4.2. This table shows the concentration of the different extracted pigments and sun-day, position in terms of latitude and longitude, and the two first statistical moments of the collected data, mean μ and variance σ^2.

Furthermore, the $R_rs(\lambda)$ spectra used as input for the QAA, the data after pre-processing, is presented in Figure. 4.6, for all the different stations with HPLC match-up. The resulting total absorption $a_{tot}(\lambda)$ is given in the same Figure.

Table 4.1.: Steps of the QAAv5 to Derive Absorption and Backscattering properties from Above-surface Remote-Sensing Reflectance R_{rs}

Step	Property	Mathematical Expression
0	$r_{rs}(\lambda)$	$\frac{R_{rs}(\lambda)}{0.52+1.7R_{rs}(\lambda)}$
1	$u(\lambda)$	$\frac{-g_0+\sqrt{(g_0)^2+4g_1\times r_{rs}(\lambda)}}{2g_1}$ $\quad g_0 = 0.089, g_1 = 0.125$
2	$a_{tot}(\lambda_0)$	$a_w(\lambda_0) + 10^{-1.146-1.366\chi-0.469\chi^2}$ $\quad \chi = \log\left(\frac{r_{rs}(443)+r_{rs}(490)}{r_{rs}(\lambda_0)+5\frac{r_{rs}(667)}{r_{rs}(490)}r_{rs}(667)}\right)$
3	$b_{bp}(\lambda_0)$	$\frac{u(\lambda_0)a_{tot}(\lambda_0)}{1-u(\lambda_0)} - b_{bw}(\lambda_0)$
4	Y	$2.0(1 - 1.2\exp(-0.9\frac{r_{rs}(443)}{r_{rs}(\lambda_0)}))$
5	$b_{bp}(\lambda)$	$b_{bp}(\lambda_0)\left(\frac{\lambda_0}{\lambda}\right)^Y$
6	$a_{tot}(\lambda)$	$\frac{(1-u(\lambda))(b_{bw}(\lambda)+b_{bp}(\lambda))}{u(\lambda)}$
7	$\zeta = \frac{a_{ph}(411)}{a_{ph}(443)}$	$0.74 + \frac{0.2}{0.8+r_{rs}(443)/r_{rs}(\lambda_0)}$
8	$\xi = \frac{a_{dg}(411)}{a_{dg}(443)}$	$\exp(S(440-410)), \quad S = 0.015 + \frac{0.002}{0.6+r_{rs}(443)/r_{rs}(\lambda_0)}$
9	$a_{dg}(\lambda)$	$\frac{(a(410)-\zeta a(\lambda))-a_w(410)+\zeta a_w(\lambda)}{\xi-\zeta}$
10	$a_{ph}(\lambda)$	$a_{tot}(\lambda) - a_{dg}(\lambda) - a_w(\lambda)$

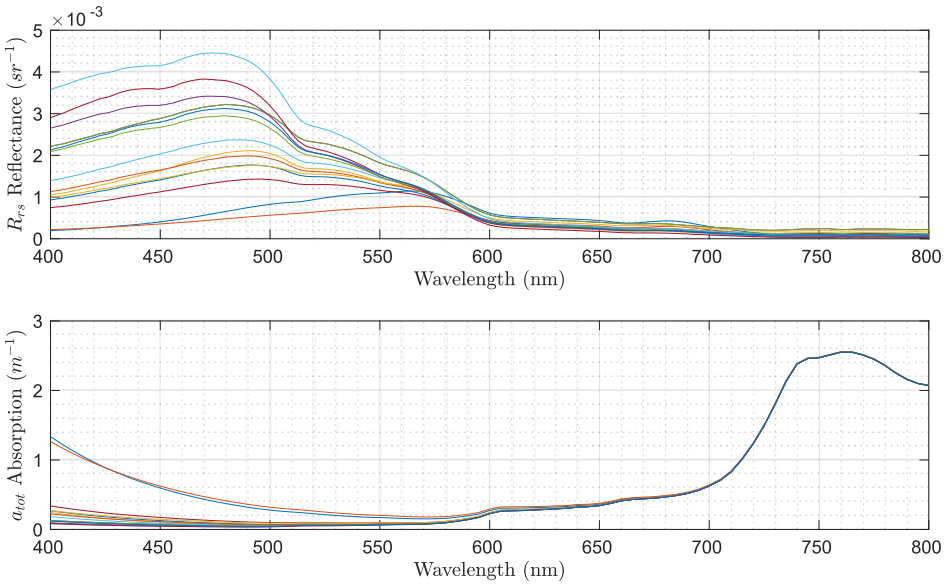


Figure 4.6.: The top plot shows the different recorded $R_{rs}(\lambda)$ median spectra for each station. These are the spectra used as input for the QAAv5. In the bottom plot the resulting $a_{tot}(\lambda)$ is given.

4.4.2 QAA Derived Results

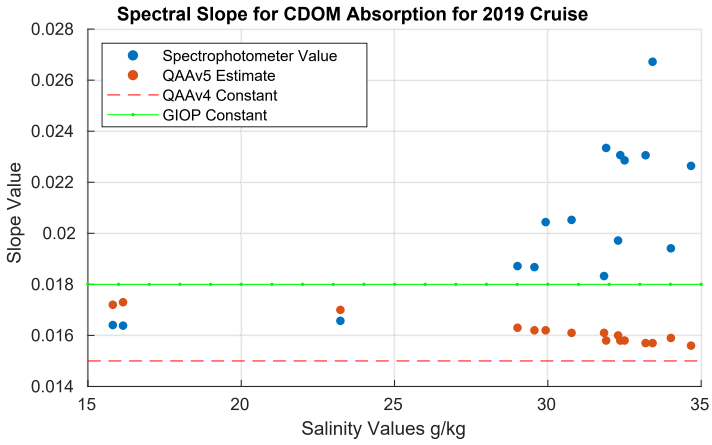
This section gives the different results from comparing the QAA with the data collected by the SP and the data analyzed using HPLC.

Table 4.2.: HPLC Data with match-ups with the SBA radiometer. Showing the concentration of the total [Chl a] content along other variables attempted to retrieved by latent variable modeling. In addition the sun day is denoted sdy, and the latitude and longitude are given as well.

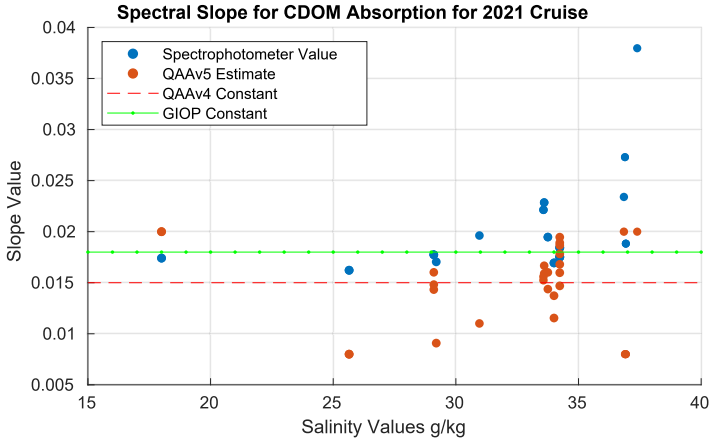
sdy	lat	lon	Chl-a	Chl-b	Chl-c	Hex-fuco	Fuco	Perid	Zea
169	5.7305	-50.983	1.212e+00	3.900e-02	1.770e-01	4.000e-02	6.110e-01	1.800e-02	4.900e-02
170	6.2209	-51.193	1.346e+00	1.400e-02	1.560e-01	1.800e-02	5.820e-01	1.000e-02	2.700e-02
171	7.9809	-53.647	2.390e-01	2.400e-02	2.400e-02	2.600e-02	1.600e-02	3.000e-03	1.470e-01
172	7.9766	-54.984	2.260e-01	9.000e-03	2.800e-02	3.400e-02	1.700e-02	3.000e-03	1.440e-01
172	7.9959	-54.978	2.070e-01	1.000e-02	2.900e-02	3.500e-02	1.900e-02	2.000e-03	1.270e-01
173	8.0000	-54.989	2.130e-01	7.000e-03	2.800e-02	3.400e-02	1.900e-02	4.000e-03	1.210e-01
174	10.823	-56.515	2.160e-01	8.000e-03	2.600e-02	3.000e-02	1.800e-02	8.000e-03	1.140e-01
178	14.269	-56.559	5.490e-01	1.180e-01	1.100e-01	1.540e-01	2.600e-02	1.400e-02	7.600e-02
179	14.607	-56.561	1.970e-01	1.600e-02	2.500e-02	3.000e-02	1.800e-02	4.000e-03	1.000e-01
182	14.012	-57.487	2.680e-01	2.100e-02	3.500e-02	3.800e-02	2.400e-02	7.000e-03	1.180e-01
183	12.015	-58.953	1.240e-01	8.000e-03	1.700e-02	1.900e-02	1.200e-02	4.000e-03	9.800e-02
184	11.99	-58.97	1.320e-01	1.000e-02	1.700e-02	2.100e-02	1.100e-02	3.000e-03	1.030e-01
186	15.184	-58.472	2.030e-01	1.200e-02	3.600e-02	3.100e-02	3.100e-02	5.000e-03	9.900e-02
187	13.832	-59.785	1.320e-01	1.000e-02	1.500e-02	1.800e-02	1.000e-02	3.000e-03	1.030e-01
188	13.185	-60.002	1.300e-01	1.100e-02	1.600e-02	1.800e-02	1.100e-02	2.000e-03	1.040e-01
μ	-	-	3.596e-01	2.113e-02	4.926e-02	3.640e-02	9.500e-02	6.000e-03	1.020e-01
σ^2	-	-	1.503e-01	7.884e-04	2.801e-03	1.117e-03	4.152e-02	2.214e-05	1.022e-03

Spectral Slope of QAA

In Figure 4.7 the S value derived from the SP are plotted alongside the QAAv5 derived results, and the QAAv4 and GIOP constants. In the figure it is possible to observe a mismatch between the derived value for S and the SP measured value for this S variable. This variable is an important factor when deriving $a_{ph}(\lambda)$ and $a_{dg}(\lambda)$.



(a) 2019



(b) 2021

Figure 4.7.: Plot of the different slopes with measured salinity values associated with the a given station along the x-axis and slope value on the y-axis for both cruises. The given slope value is used to determine the effects from CDOM.

Comparison with HPLC

The HPLC data is only available for the 2019 cruise at the time of writing. Figure 4.8 shows the correlation and scatter plots for different wavelengths, namely 443, 481, 618 and 665nm, with Chl-a concentrations measured by HPLC. These are selected as they are strongly correlated with high absorption due to high Chl-a concentrations. The wavelengths for the high absorption peaks are taken from [136].

The QAA produce a meaningful spectral shape for the $a_{tot}(\lambda)$, i.e. there is a strong correlation between the measured absorption peaks in the spectra and the absorption peaks associated with high absorption with higher concentrations of Chl-a. However, the derived absorption characteristics for $a_{ph}(\lambda)$ does not correlate well with Chl-a, as seen in Figure 4.8. This indicates that the spectra derived from $a_{tot}(\lambda)$ to $a_{ph}(\lambda)$ in QAA does not correlate well with the values found through HPLC, which is considered to be accurate [136].

Determining $a_{ph}(\lambda)$ from $a_{tot}(\lambda)$ using the QAA is effected by the choice of reference wavelength, as well as how the spectral slope S for CDOM absorption is computed. In the original version of QAA [135, 138] a constant is used for S. QAAv5 attempts to compute the S value [142].

However, it should be noted that according to [135], step 8 of Table 4.1 is of secondary importance. As different SAAs vary how they compute S, and it is stated to be of lesser importance, its impact on the derivation of $a_{ph}(\lambda)$ could be less significant. The loss in terms of correlation with chl-a concentrations from HPLC, shown in Figure 4.8, indicates that there may be some undesired effects of the derivation not accounted for when using the QAA approach.

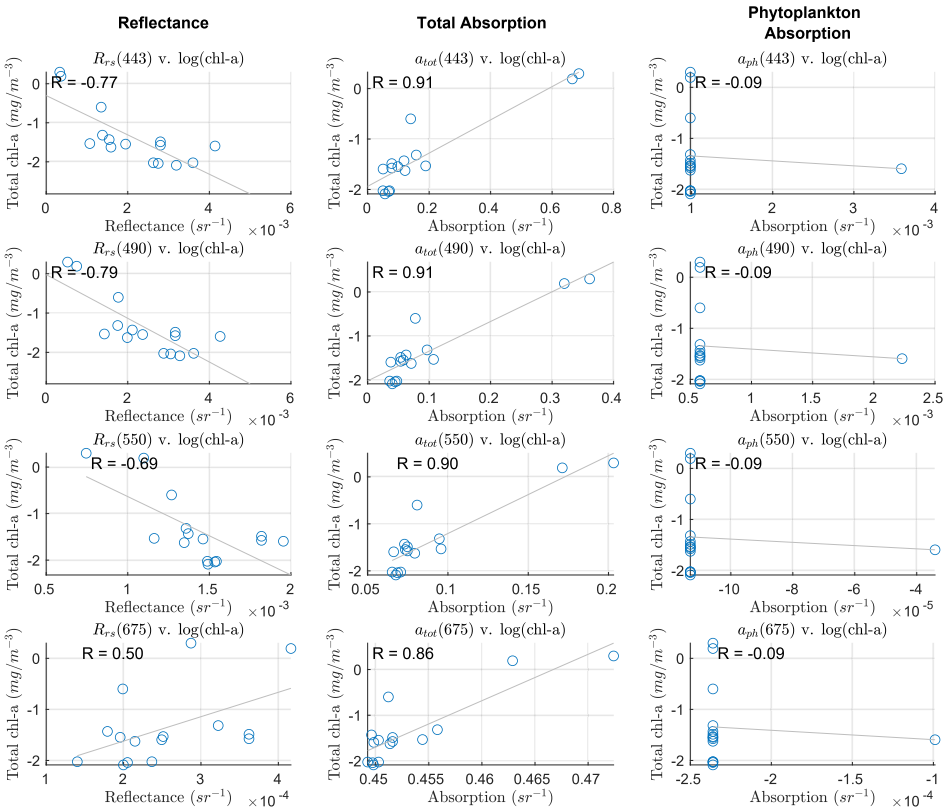


Figure 4.8.: This figures shows in different columns, left to right, reflectance, total absorption and estimated phytoplankton absorption and how these different spectra correlate with wavelengths closest to significant absorption peaks. The absorption peaks are selected using [136] for R_{rs}, a_{tot}, and a_{ph}. The wavelengths are, from top to bottom, 443, 490, 550 and 675. These are selected as they are strongly correlated with high absorption due to high Chl-a concentrations.

Comparison With Spectrophotometer

The plots for comparing the QAA derived results from the SBA radiometer with the results from the SP is shown in 4.9 and 4.10, for the 2019 and 2021 cruise, respectively. As can be seen from these plots, there are some relatively solid correlations for the selected wavelengths, especially in the 400 to 550-nanometer region. The relationship is not as substantial for the absorption spectra for phytoplankton pigments. The absorption due to CDOM seems to be derived well for the shorter wavelengths for both data sets. The total absorption might suffer from the same discrepancies as to the absorption due to phytoplankton pigments.

Even though it is not shown here, it should be noted that the SP data related to the 2021 cruise is of better quality than that collected for the 2019 cruise.

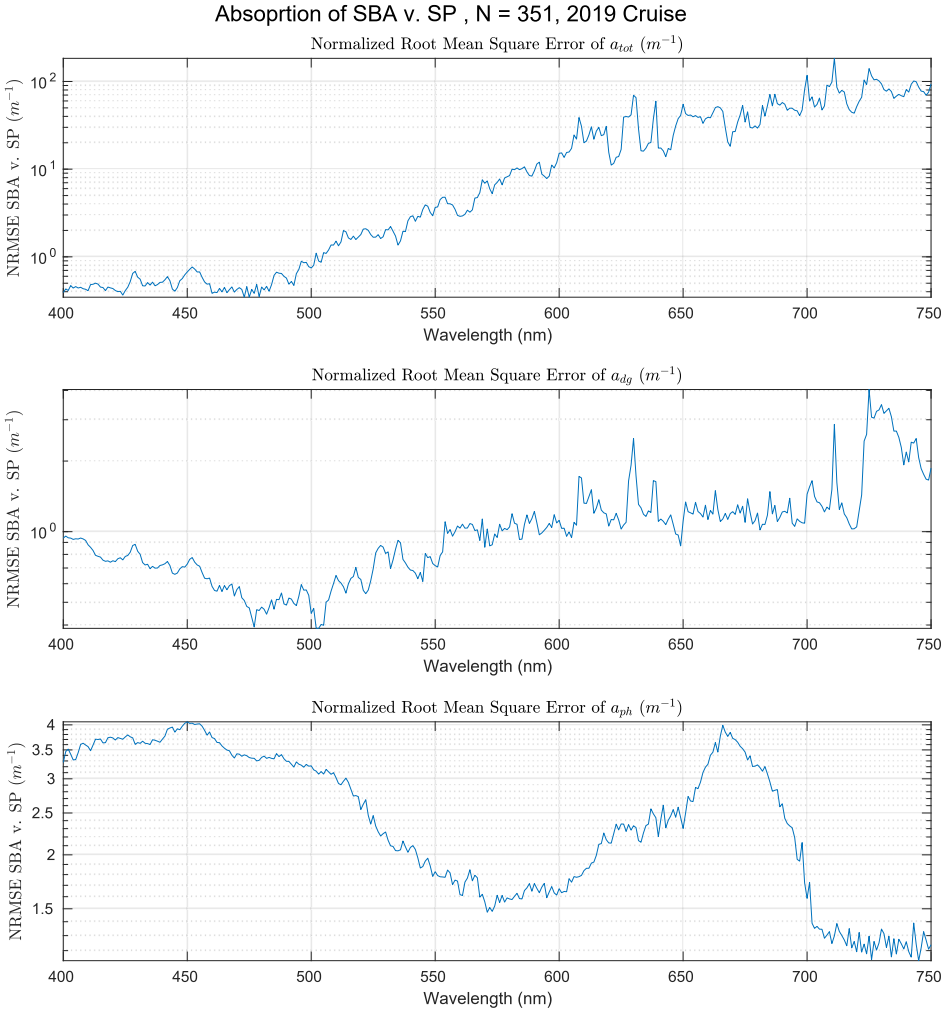


Figure 4.9.: Results for 2019 Cruise. The figure shows a comparison between results from the SP and results from the radiometer of different absorption spectra at different wavelengths. SBA is the abbreviation for the radiometer, while SP is the abbreviation for the spectrophotometer. N is the number of correlated samples for each subplot.

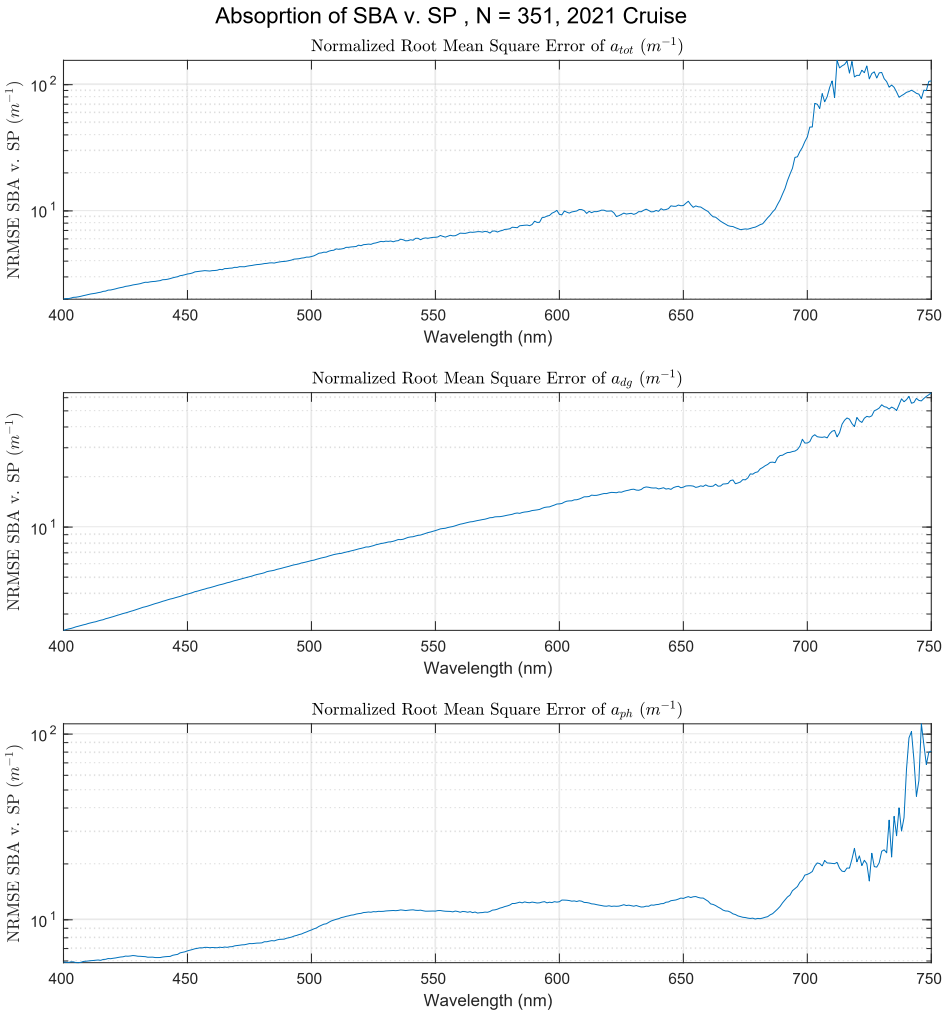


Figure 4.10.: Results for 2021 Cruise. The figure shows a comparison between results from the SP and results from the radiometer of different absorption spectra at different wavelengths. SBA is the abbreviation for the radiometer, while SP is the abbreviation for the spectrophotometer. N is the number of correlated samples for each subplot.

4.4.3 Limitation

The data set used in the analysis presented here is limited. As with most similar studies, the results given here would be strengthened with more data.

The fidelity of the measured spectra from the SBA radiometer and the QAA derived results are to some extent attempted mitigated by the data preparation discussed in section 4.3.1. Conversely, the fidelity of the data collected from the HPLC and SP has not been under the same level of scrutiny. These data sources are prone to have errors as well.

4.4.4 Discussion

In the observed waters, CDOM dominates in some of the parts of the VIS-NIR spectra that are commonly used for estimation of [Chl a] concentration [16, 139], especially at wavelengths such as 443 nm, which is a cornerstone for many band-ratio algorithms [16].

The 65 band-ratio algorithms rigorously investigated by [16], with a suitable validation scheme, achieved at best an R^2 score of 0.8601, which was obtained for the multi-spectral OCTS sensor with a total of 12 bands. The result was obtained using the reflectance data from 443, 490, 516, and 565 nm. These four wavelengths, while being good predictors for the Chl-a in the open ocean, may suffer from the interference contributed by the presence of other optically active substances, e.g., CDOM [139]. The validation scheme for the band-ratio algorithms, described as algorithm tuning in the original paper, optimized the regression coefficients to achieve a maximum R^2 while maintaining a slope of 1 and an intercept of 0.

There is more information to be retrieved in a finer spectral resolution [16, 130, 136, 139], and the argument that multispectral band-ratio algorithms does not fully utilize this information is made in [130] and [16, 130].

While many of the sensors with higher spectral resolution exhibit higher noise levels per band, this is not the same as saying that the data is contaminated by more noise. Any information can be good information, as long as the uncertainties in the measurements are correctly understood [143]. Thus, data points strongly correlated, either spatially, spectrally, or both, can give better inference than a single coarse measurement with good noise characteristics. With an adequate characterization of the noise in a given hyperspectral sensor, these artifacts can compensate for potentially yielding an overall better performance.

The QAAv5 derived total absorption $a_{tot}(\lambda)$ spectra correlates well with the HPLC measured Chl-a concentration. univariate correlations for wavelengths that are active [136], shown in Figure 4.8, for Chl-a shows potential for inferring the Chl-a concentration with a similar or better performance as to what is found in [16]. As the spectral resolution increases, it becomes possible to discriminate between IOPs and the different signal contributions. Given the results presented here, these properties seem to hold even for optically complex waters like the Amazon river plume.

4.5 Conclusions

The QAA, while being a compelling approach in determining IOPs, can benefit further by adjusting the last steps of the algorithm. The QAA does not use any spectral models of pigment and CDOM absorption. Thus, the algorithm does not have many empirical assumptions that need to be determined by an operator or similar. The QAA instead derive these properties from the a_{tot}, and not the other way around. The spectral slope S is an important factor in determining the a_{dg} ad a_{ph} from a_{tot}. Thus, discrepancies between the S measured by the SP and the S computed by the QAAv5, as shown in Figure 4.7, warrants further investigation.

Furthermore, some of the intermediate derived properties contain non-physically valid values. Whether this could best be resolved by revisiting the input data, reviewing the current steps of the QAAv5 or shadow corrected QAA, or adding regularization terms to the QAA is unclear. As the QAA was originally developed for multispectral sensors, but the results found could be extended to the hyperspectral domain, it could be interesting to perform the same type of model development using a higher and more detailed spectral resolution and possibly more wavelengths as part of the model.

With a finer spectral resolution, the accuracy of Chl-a inference from properties derived from hyperspectral remote sensing can be improved. Even without the latter stages of the QAA producing the expected relationships, some strong correlation between the remotely sensed hyperspectral data collected by the radiometer and the Chl-a measured by HPLC. For wavelengths known to be useful indicators of Chl-a concentration, a high correlation coefficient R^2 for the a_{tot} was found, without any specific pre-processing for the given region. While a_{tot} seems to correspond well with other measurements, the QAA does not seem to contribute the absorption effects correctly to the other derived adsorption spectra, here presented by a_{dg} for CDOM and a_{ph} for phytoplankton absorption.

The QAA can in some cases be used successfully on hyperpsectral data rertirved from a SBA radiometer used in the Amazon River plume, as can be seen from the absorption plots given in Figure 4.9 and 4.10. This suggests that the QAA can derive certain parts of the absorption spectra in these complex water types investigated here, i.e., the Amazon River Plume. The QAA seem to be specifically well suited to derive the absorption properties due to CDOM absorption for the shorter wavelengths, e.g., 400 to 550 nanometer.

In summary, these preliminary results show that the QAA approach is suited to derive some IOPs for optically complex waters such as the Amazon River plume. However, there are significant discrepancies, especially concerning the derived absorption spectra related to phytoplankton pigment absorption. This warrants further investigation, where how the computation of the spectral slope S is conducted, given Figure 4.7 could be one potential path. With improved performance in the longer wavelengths and the latter derived absorption spectra, the QAA or other GIOP variants could provide great value and insight to more complex water types in the future.

Chapter 5

Dimensionality Reduction and Target Detection

Strive not to be a success, but rather to be of value.

Albert Einstein

Target detection is a popular application of hyperspectral remote sensing images. It is also an approach that is intended to be used in the HYPSO-1 satellite to allow for higher throughput of relevant information from oceanographic observations. By only sending down information detailing what kind of targets were detected during a capture, the data volume can be significantly reduced as discussed in Section 2.2. To enhance the detection rate, it is common to do preprocessing to reduce the effects of noise and other undesired interference with the observed spectral signatures. In current Earth observing systems, in particular, small satellite systems such as HYPSO-1, data rate limitations can make the utilization of sensors with high spectral dimensionality undesirable and even unobtainable due to the high data volume.

This chapter shows the effect of different, frequently used dimensionality reduction and noise removal methods on multiple classical methods for signature matched target detection often used in hyperspectral imaging. The dimensionality reduction methods used here differ from spectral resampling or band selection in the sense that the original spectral resolution can be restored, with some loss, via a linear transformation.

This chapter provides an investigation on the effects of combining dimensionality reduction and target detection. We demonstrate that the resulting data cube has reduced dimensionality and suppressed undesired effects. The ability to correctly detect spectral phenomena has improved for the data sets used here while also achieving a reduced data volume. Combining dimensionality reduction and target detection can also reduce the number of computational operations needed in the latter stages of processing when operating on the sub-space, as discussed in chapter 2 and 3.

The observed effects are demonstrated by using simulated and real-world hyperspectral scenes. The real-world scenes are from well-calibrated sensors of classified agricultural and urban areas, e.g., AVIRIS, ROSIS, and Hyperion. These scenes are frequently used when investigating target detection performance. The simulated scenes is generated using the ASTER library.

5.1 Motivation

The effect of dimensionality reduction as a method for compression, and in turn, on target detection is investigated. The effectiveness of lossy compression algorithms has been evaluated by comparing the restored scene with the original scene [144]. This approach is limited because the original scene cannot be void of noise or other unwanted artifacts from the sensor, which does not give a definitive answer to whether the lossy compression algorithms reduce noise or degrade the information in the signal. This chapter discusses the effect of dimensionality reduction on signature-based target detection for hyperspectral remote sensing to answer parts of this question. This approach does not quantify the ability to restore the original data, but rather quantify the effect of the lossy compression on signature-based target detection, both through the F_1-score [145] for target detection performance and visibility as a metric for target detection robustness [146].

The results show that a high compression ratio can be achieved by exploiting structures found in the first- and second-order statistical moments. Given the application of target detection, the projected subspace of the data provides, for most cases, improved performance for both the F_1-Score and the visibility. As the data can be analyzed in its compressed state, this can in turn significantly reduce the number of operations required to perform further or more advanced onboard processing in a smallsat system, while at the same time having reduced the total data volume that needs to be stored and transmitted, as discussed in Section 2.2 as well.

5.1.1 Notation

All hyperspectral images are inherently three-dimensional cubes with two spatial dimensions and one spectral dimension. However, in all the subsequent sections a two dimensional matrix representation of the hyperspectral data is used, and the relationship is given in equation (5.1) and (5.2), where each $\mathbf{x}_{i,j}$, and subsequently $\mathbf{x}_i$, is a vector of the spectra with a sub-script re-indexation from equation (5.2) to equation (5.3). Here the original data set, prior to any dimensionality reduction, is given as $\mathbf{X}_o$ from

$$\mathbf{X}_{Cube} = \begin{bmatrix} \mathbf{x_{1,1}} & \mathbf{x_{1,2}} & \cdots & \mathbf{x_{1,i}} \\ \mathbf{x_{2,1}} & \mathbf{x_{2,2}} & \cdots & \mathbf{x_{2,i}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x_{j,1}} & \mathbf{x_{j,2}} & \cdots & \mathbf{x_{j,i}} \end{bmatrix} \tag{5.1}$$

$$\mathbf{X}_{Matrix} = \begin{bmatrix} \mathbf{x_{1,1}} & \cdots & \mathbf{x_{1,i}} & \mathbf{x_{2,1}} & \cdots & \mathbf{x_{2,i}} & \cdots & \mathbf{x_{j,1}} \cdots & \mathbf{x_{j,i}} \end{bmatrix} \tag{5.2}$$

$$\mathbf{X}_o = \Big[\mathbf{x}_1, \quad \mathbf{x}_2, \quad \mathbf{x}_3, \quad \cdots, \quad \mathbf{x}_i, \quad \cdots, \quad \mathbf{x}_N\Big] \tag{5.3}$$

.

5.2 Background

This section briefly describes the methods and algorithms used in the analysis. An in-depth description of the theory applied can be found in the references [4].

5.2.1 Dimensionality Reduction

High dimensional data such as a hyperspectral image can be used in classification or target detection applications. The high-dimensional data contains more observed variables than potential classes to classify or targets to detect in the image; This high number of variables is a source of variation and redundancy in hyperspectral images [1, 4]. Some of the information in the image is thus redundant for analysis due to a strong correlation between certain bands, e.g., neighboring bands. Dimensionality reduction is a common approach to handle the high dimensionality in hyperspectral imaging. In the analysis that follows, four classic methods for dimensionality reduction are explored and analyzed in the context of target detection.

Principal Component Analysis (PCA) is a method to reduce the dimensionality by utilizing the correlation of different variables in a set. [147] Within PCA the principal components are ordered according to variance, such that the first components carry more information concerning the full data space than the later components. PCA is an optimal method for noise filtering if the variables or spectral signal under observation carry additive independent white noise.

The eigenvalues σ^2 corresponding to the principal components are computed as follows

$$\begin{aligned} \mathbf{m} &= \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \\ \Sigma_o &= \frac{1}{N}\mathbf{X}_o\mathbf{X}_o^T - \mathbf{m}\mathbf{m}^T \\ \det(\Sigma_o - \sigma^2\mathbf{I}) &= 0. \end{aligned} \tag{5.4}$$

In the above equation, Σ_o is the covariance matrix computed from $\mathbf{X}_o$, $\mathbf{I}$ is an identity matrix with dimensions equal to the number of spectral channels, and σ^2 is an eigenvalue. Then, the corresponding unit eigenvectors V_{PCA} are ordered according to the magnitude of the eigenvalues, such that

$$\Sigma_o\mathbf{V}_{PCA} = \mathbf{V}_{PCA}\mathbf{D} \tag{5.5}$$

with $\mathbf{D}$ is a diagonal matrix consisting of the eigenvalues, ordered according to the magnitude with the highest eigenvalues as the first element, and $\mathbf{V}_{PCA}$ is a unitary matrix consisting of the eigenvectors to fulfill $\Sigma_o \mathbf{v}_j = \sigma_j^2 \mathbf{v}_j$. From this, the transformed subspace can be represented with the columns of $\mathbf{Z}_{PCA}$ as basis vectors, where the dimensionality reduction is performed by selecting a subset of the eigenvectors related to the largest eigenvalues, as

$$\mathbf{Z}_{PCA} = \mathbf{V}_{PCA}^T \mathbf{X}_o, \tag{5.6}$$

with $\mathbf{X}_o$ defined in equation (5.3), a vector of a hyperspectral signal or a matrix of the observed data with pixels in each column and spectral information in the rows.

Maximum Noise Factoring (MNF) is a dimensionality reduction method that attempts to account for the source of the noise. Whereas PCA is only dependent on the variance in the data, MNF sorts the principal components based on the estimated signal-to-noise properties. The underlying assumption made in the MNF transform is that the noise is additive, but not normally distributed white noise, expressed as

$$\Sigma_o = \Sigma_s + \Sigma_n \tag{5.7}$$

with Σ_s as the covariance of the signal and Σ_n is the covariance of the noise. As the MNF seeks to maximize the eigenvalues for Signal-to-Noise Ratio (SNR) ratio and decorrelate the covariance matrix, it can be performed as a two-step PCA. In the analysis given here, two different approaches for noise estimation were used. In equation (5.8), the noise vector set $\mathbf{X}_n$ is computed by taking the difference between neighboring pixels in a spatial area of uniform spectral content [148], later denoted as GMNF. The GMNF attempts to say something about the noise characteristics of spectra using the underlying assumption that pixels of the same material will have very similar spectra, and the difference is mainly due to the noise properties of the sensor [4]. In equation (5.9) the noise vector set $\mathbf{X}_n$ is computed by taking the difference between neighboring data points in the spectral dimension projected into the space of the spectral band in used [4, 147]. This method is denoted as BMNF. The underlying idea behind this method is to exploit the spectral correlation of the different parts of the spectra. This can be expressed as

$$\mathbf{X}_n = \begin{bmatrix} \mathbf{x}_1 - \mathbf{x}_2, & \mathbf{x}_2 - \mathbf{x}_3, & \cdots, & \mathbf{x}_{N-1} - \mathbf{x}_N \end{bmatrix} = \begin{bmatrix} \hat{\mathbf{n}}_1, & \hat{\mathbf{n}}_2, & \cdots, & \hat{\mathbf{n}}_{N-1} \end{bmatrix} \tag{5.8}$$

$$
\begin{aligned}
\mathbf{Y} &= \mathbf{X}_o^T \\
\mathbf{Y}_{(-k)} &= \begin{bmatrix} \mathbf{y}_1, & \mathbf{y}_2, & \cdots, & \mathbf{y}_{k-1}, & \mathbf{y}_{k+1}, & \cdots, & \mathbf{y}_L \end{bmatrix} \\
\tilde{\mathbf{n}}_k &= \mathbf{y}_{(k)} - \mathbf{Y}_{(-k)} \left(\mathbf{Y}_{(-k)}^T \mathbf{Y}_{(-k)} \right)^{-1} \mathbf{Y}_{(-k)}^T \mathbf{y}_{(k)} \\
\mathbf{X}_n^T &= \begin{bmatrix} \tilde{\mathbf{n}}_1, & \tilde{\mathbf{n}}_2, & \cdots, & \tilde{\mathbf{n}}_L \end{bmatrix}
\end{aligned}, \tag{5.9}
$$

with L as the number of spectral channels, $\mathbf{X}_n$ is in turn used to compute the noise covariance matrix Σ_n from equation (5.10). The singular value decomposition is used for numerical stability and reduced computational time for data sets with more samples than variables. [149] The singular value decomposition is given in this chapter with matrices $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$, with $\mathbf{S}$ being a diagonal matrix with entries sorted according to the magnitude of the eigenvalue, $\mathbf{V}$ being an orthogonal square matrix with dimensions equal to the number of spectral channels, and $\mathbf{U}$ being an orthogonal square matrix with dimensions equal to the number of pixels. With $\mathbf{X}_o$ as the original data set and $\mathbf{X}_w$ as the noise-whitened data set, where the the noise-whitening transformation with a whitening matrix expressed in equation (5.10) as $\mathbf{U}_n\mathbf{S}_n^{-1/2}$, yields the whitened random vector $\mathbf{X}_w$ with unit diagonal covariance and zero mean, further expressed as follows

$$
\Sigma_n = \mathbf{U}_n\mathbf{S}_n\mathbf{V}_n^T \qquad \mathbf{X}_w = \mathbf{X}_o\mathbf{U}_n\mathbf{S}_n^{-1/2} = \mathbf{U}_w\mathbf{S}_w\mathbf{V}_w^T. \tag{5.10}
$$

From this, the transformation matrix $\mathbf{V}_{MNF}$ can be expressed as

$$
\mathbf{V}_{MNF} = \mathbf{U}_n\mathbf{S}_n^{-1/2}\mathbf{V}_w. \tag{5.11}
$$

Hence, the transformed subspace can be represented as $\mathbf{Z}_{MNF}$, where the dimensionality reduction is performed by selecting a subset of the singular vector corresponding to the most significant singular value, in the following way. With

$$
\mathbf{Z}_{MNF} = \mathbf{V}_{MNF}^T\mathbf{X}_0, \tag{5.12}
$$

where $\mathbf{X}$ is a vector or matrix to be projected into the subspace.

Independent Component Analysis (ICA) attempts to decompose a multivariate signal into independent non-Gaussian signals, i.e. a decomposition that provides statistical independence between the estimated components. Blind source separation, or ICA, of a mixed-signal, can separate the different signal sources well when the statistical independence assumption is correct. While PCA and MNF are computed based on first and second-order statistical moments when estimating the subspace, ICA utilizes higher-order statistical moments. The underlying assumption can conversely be stated as

$$\mathbf{Z}_{ICA} = \mathbf{V}_{ICA}^{-1}\mathbf{X}_o, \tag{5.13}$$

where $\mathbf{V}_{ICA}^{-1}$ is the separating matrix with the pre-selected number of components or assumed some unique spectral signatures, and $\mathbf{Z}_{ICA}$ is the transformed data set with values per pixel corresponding to the computed abundance of a derived spectral signature. In $\mathbf{V}_{ICA}^{-1}$ every row can be regarded as an independent signal, e.g., the spectral signature of a material. The separating matrix $\mathbf{V}_{ICA}^{-1}$ is computed iteratively over the entire data set. In this chapter, Joint Approximation Diagonalization of Eigen-matrices (JADE) is used to compute the components or independent signals, due to its convergence properties when compared with other approaches [150]. It should be noted that for computation of the components, as opposed to using a pre-computed projection matrix from a relevant dataset, the JADE ICA approach can be more computationally demanding than other approaches such as FastICA [150, 151]. JADE ICA separates observed mixed signals into latent source signals, i.e. underlying structures not directly observed, by exploiting kurtosis, the fourth-order statistical moment, computed as shown in equation (5.14). Kurtosis is a measure of how Gaussian the observed data distribution is, and is used for defining independence between the source signals. Thus, JADE ICA seeks an orthogonal rotation of the observed mixed vectors to estimate source vectors which possess high values of excess kurtosis. Kurtosis is defined as

$$\mathbf{Kurt}[\mathbf{X}] = \mathbf{E}\Big[\Big(\frac{\mathbf{X}-\mu}{\sigma}\Big)^2\Big]. \tag{5.14}$$

5.2.2 Target Detection

Furthermore, in this chapter, four classical signature-based target detection algorithms are used in the analysis [4, 152]. This subset of target detection algorithms exploits a priori information about the desired target. There is an assumption that the target signature is available and has been normalized to fit the scene. These target detection algorithms can also be considered as linear processes that utilize the information provided by first and second-order statistics [4, 153].

More advanced nonlinear target detection methods can exploit the statistical moments of a higher order than traditional methods. This may, in some cases, yield better performance, but it demands a higher level of a priori knowledge of a specific application and more training data to be able to map the solution space, and can be prone to such pitfalls as overfitting. This makes it difficult to claim whether a given target detection algorithm is superior to other alternatives [154]. Thus, the use of classical target detection algorithms may be sufficient for a given application, both in terms of robustness and target detection performance.

Spectral Angle Mapper (SAM) is a detection algorithm based on the following hypotheses

$$\mathbf{H}_0 : \mathbf{x} = \mathbf{b} \qquad \mathbf{H}_1 : \mathbf{x} = \alpha\mathbf{s} + \mathbf{b}, \tag{5.15}$$

where $\mathbf{x}$ is a vector in the data set, $\mathbf{b}$ is the background and clutter noise, α is a parameter accounting for uncertainty in the desired signal strength, and $\mathbf{s}$ is the desired signal. The SAM operator can be expressed as in equation (5.16) and a derivation of the operator can be found in [4].

$$r_{SAM}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{x})^2}{(\mathbf{s}^T\mathbf{s})(\mathbf{x}^T\mathbf{x})} \tag{5.16}$$

Adaptive Cosine Estimator (ACE) is a detection algorithm based on the following hypotheses

$$\mathbf{H}_0 : \mathbf{x} = \mathbf{b} \qquad \mathbf{H}_1 : \mathbf{x} = \alpha\mathbf{s} + \beta\mathbf{b}, \tag{5.17}$$

where $\mathbf{x}$ is a vector in the data set, $\mathbf{b}$ is the background and clutter noise, $\mathbf{s}$ is the desired target signal, and α and β is parameters accounting for uncertainty in the signal strengths. The ACE operator can be expressed as

$$r_{ACE}(\mathbf{x}) = \frac{(\mathbf{s}^T\Sigma_o^{-1}\mathbf{x})^2}{(\mathbf{s}^T\Sigma_o^{-1}\mathbf{s})(\mathbf{x}^T\Sigma_o^{-1}\mathbf{x})}. \tag{5.18}$$

Constrained Energy Minimization (CEM) is a detection algorithm based on minimization of the projected background energy, where the energy can be expressed as in equation (5.19) leading to the expression found in equation (5.20), with the solution found in equation (5.21). A derivation of the operator can be found in the references [4].

$$E = \frac{1}{N}\sum_{i=1}^{N}\mathbf{h}^T\mathbf{x}_i{\mathbf{x_i}}^T\mathbf{h} = \mathbf{h}^T\mathbf{R}_o\mathbf{h} \qquad \mathbf{R}_o = \frac{1}{N}\mathbf{X}_o\mathbf{X}_o^T \tag{5.19}$$

$$\min_{\mathbf{h}} \mathbf{h}^T\mathbf{R}_o\mathbf{h} \qquad \text{s.t.} \quad \mathbf{h}^T\mathbf{s} = 1 \tag{5.20}$$

$$r_{CEM}(\mathbf{x}) = \frac{(\mathbf{s}^T\mathbf{R}_o^{-1}\mathbf{x})^2}{(\mathbf{s}^T\mathbf{R}_o^{-1}\mathbf{s})} \tag{5.21}$$

Orthogonal Subspace Projection (OSP) is a detection algorithm based on the hypotheses given as

$$\mathbf{H}_0 : \mathbf{x} = \mathbf{B}\beta + \mathbf{n} \qquad \mathbf{H}_1 : \mathbf{x} = \alpha\mathbf{s} + \mathbf{B}\beta + \mathbf{n}, \tag{5.22}$$

where $\mathbf{x}$ is a vector in the data set, $\mathbf{B}$ represent the background subspace, β is the background basis coefficient vector, $\mathbf{n}$ is independent normally distributed noise with zero mean, α is a parameter accounting for uncertainty in the desired signal strength, and $\mathbf{s}$ is the desired target signal. The OSP operator can be expressed as in equation (5.23) and a derivation of the operator can be found in the references [4]. In this chapter, the N-FINDR endmember extraction algorithm as described in [155]

is used to estimate the background subspace $\mathbf{B}$, to separate it from the desired signal. The desired signal is identified as the spectral signature closest to the sought-after signature based on SAM and removed to make up the background subspace $\mathbf{B}$.

$$r_{OSP}(\mathbf{x}) = \mathbf{s}^T(\mathbf{I} - \mathbf{B}(\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T)\mathbf{x} = \mathbf{s}^T\mathbf{P}_{\mathbf{B}}^{\perp}\mathbf{x} \tag{5.23}$$

5.3 Data Set Description

An overview of the real-world and simulated data sets used in the analysis is given. All real-world scenes are publicly available [156]. A short description of how the simulated data sets are generated is also provided. True color images for all the real-world data sets are given in Figure 5.1, with the image showing the distribution of endmembers.

Gray space represents unclassified pixels but is still used in model creation. In addition, all performance metrics are computed on the entire scene. For each scene used in the analysis, Table 5.1 gives the sensor, the spatial extent of the pixels, the number of classes in each set, the spectral range, and the resolution of each sensor. Furthermore, a short description of the characteristics of each data set is given, and the different compression ratios for dimensionality reduction are given in Table 5.2. As training samples for all the dimensionality reduction methods and target detection methods, 30% of the pixels in a given data set were randomly selected as training samples. No manually selected portion of the scene was used to compute $\mathbf{X}_n$, for either of the MNF transforms. The poorer noise estimation in the Pavia data set is probably due to the in-homogeneity of the scene, compared to the more homogeneous scenes e.g. Salinas.

None of the pixels in the data sets presented here are explicitly mutually exclusive, i.e. there can be more than one material in a given pixel.

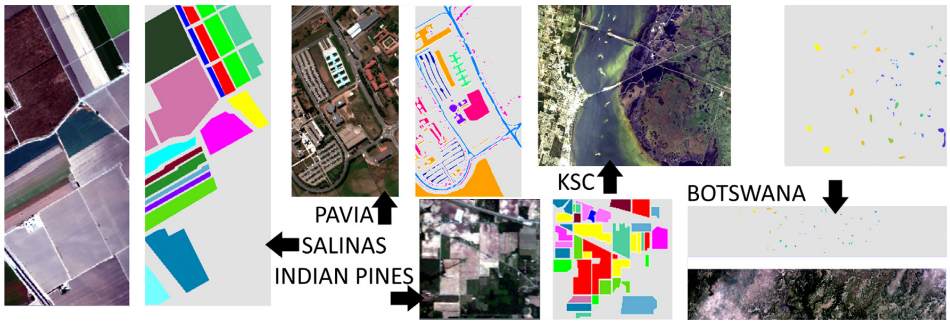


Figure 5.1.: True color images for all the real world data sets The accompanying image shows the distribution of endmembers. Grey areas is unclassified pixels, which is still used in model creation and performance testing.

The Indian Pines scene was gathered by the Airborne Visible / Infrared Imaging Spectrometer (AVIRIS) sensor with 224 spectral reflectance bands. The test site is in North-western Indiana and consists of 145 by 145 pixels. The scene is a subset of a larger one and contains two-thirds agriculture and one-third natural perennial vegetation e.g. forest [156, 157].

The Salinas scene was collected by the AVIRIS sensor over Salinas Valley, California. The scene has a spatial resolution of 3.7-meter pixels, which can be considered a high spatial resolution. The area consists of 512 by 217 pixels and includes vegetables, bare soils, and vineyard fields [156, 158].

The Pavia Centre Scene is acquired by Reflective Optics System Imaging Spectrometer (ROSIS) during a flight campaign over Pavia in northern Italy. The scene has a spatial resolution of 1.3-meter pixels, and 102 of the 115 spectral reflectance bands in the wavelength range of $430nm$ to $860nm$ are made available in the data set. The other bands were removed due to high absorption in the atmosphere. [156, 159]

The Kennedy Space Centre Scene (KSC) is a site in Florida. The scene has a spatial resolution of 18-meter pixels. The publicly available data set has removed some bands to mitigate the effects of bad detectors, calibration errors, and other anomalies. The subset consists of 176 of the 224 bands, and the classified pixels provided represent various land cover types that occur. [156, 158]

The Botswana scene is acquired using Hyperion, a payload on the NASA EO-1 satellite. The acquired scene is from the Okavango Delta. The Hyperion sensor on EO-1 had a spatial resolution of 30-meter pixels and 224 spectral reflectance bands. The publicly available data set has removed bands to mitigate the effects of bad detectors, calibration errors, and other anomalies, and the remaining subset consists of 145 of the 224 bands available. [156, 160]

Simulated Synthetic Hyperspectral Scenes

The ASTER spectral library [161] was used as a basis for the spectral signatures in the simulated scene. Ten spectral signatures were selected from the spectral library to make up the different synthetically generated images. The spatial scene was created using *Hyperspectral Imagery Synthesis toolbox* [162], using the Matern covariance function with $\theta_1 = 0.5$ and $\theta_2 = 0.5$, and the spectral signatures to generate noise-free images with a reasonable spatial distribution of the

Table 5.1.: The data sets used in the analysis and important parameters.

Data set	Sensor	Image format	Endmembers	Range (nm)	Resolution (nm)	Reference
Indian Pines	AVIRIS	145x145	16	[400,2500]	10	[157, 158]
Salinas	AVIRIS	512x217	16	[400,2500]	10	[158]
Pavia	ROSIS-3	610x340	9	[430, 860]	4	[159]
KSC	AVIRIS	512x614	13	[400,2500]	10	[158]
Botswana	Hyperion	1476x256	14	[400, 2500]	10	[160]
Simulated Scenes	ASTER	100x100	10	[400,925]	4	[161]

materials. 10 different synthetic scenes were generated and used to get the average performance statistics.

Noise is added to the synthetic scenes to simulate undesired effects and artifacts according to the following relationship [147]

$$\mathbf{y}_i = \mathbf{x}_i + \mathbf{n}_i, \tag{5.24}$$

where $\mathbf{x}_i$ is the spectral signature taken from the ASTER spectral library and $\mathbf{n}_i$ is subject to

$$SNR \equiv 10 \log_{10} \frac{\mathbb{E}[\mathbf{x}_i^T \mathbf{x}_i]}{\mathbb{E}[\mathbf{n}_i^T \mathbf{n}_i]}, \tag{5.25}$$

where the SNR is set to 50 and the variance of the noise signal $\mathbf{n}_i$, as a function of spectral channel i, is given as

$$\sigma_i^2 = \sigma^2 \frac{e^{-\frac{(i-L/2)^2}{2\eta^2}}}{\sum_{j=1}^{L} e^{-\frac{(j-L/2)^2}{2\eta^2}}} \tag{5.26}$$

Where L is the number of spectral channels, which gives a $\sigma^2 \approx 9.73 \times 10^{-07}$ and a σ_i as shown in figure 5.9, with $\eta = 18$. Thus the generated noise signal have different characteristics for different wavelengths, making it colored noise rather than white noise.

5.4 Methods

The analysis performed in this chapter differs from earlier comparisons of dimensionality reduction and target detection performance [144, 146, 152, 163]. Rather than using the receiver operating characteristic (ROC) curve [145] to determine the performance, a combination of the F_1-score [145] and visibility [146] is used.

The original data sets [156] have not been processed prior to dimensionality reduction or target detection. Earlier publications [57, 158] have stated that it is not necessary to calibrate scenes captured with AVIRIS to reflectance before applying hyperspectral processing methods due to the

Table 5.2.: Compression ratios for different number of components after dimensionality reduction. The compression ratio is only dependent on number of components. Computed as CR = spectral channels / components

Data set \Components	50	45	40	35	30	25	20	15	10	5
Indian Pines	4.48	4.98	5.60	6.40	7.50	8.96	11.20	14.93	22.40	44.80
Salinas	4.48	4.98	5.60	6.40	7.50	8.96	11.20	14.93	22.40	44.80
Pavia	2.06	2.29	2.58	2.94	3.43	4.12	5.15	6.87	10.30	20.60
KSC	3.52	3.91	4.40	5.03	5.87	7.04	8.80	11.73	17.60	35.20
Botswana	2.90	3.22	3.63	4.14	4.83	5.80	7.25	9.67	14.50	29.00
Simulated Scenes	2.40	2.67	3.00	3.43	4.00	4.80	6.00	8.00	12.00	24.00

high SNR of the sensor and a relatively flat quantum efficiency function i.e. a similar sensitivity over the captured electromagnetic spectrum. Following this approach, the combination of DR and TD presented in this chapter is trained on radiance spectra and operates directly on each pixel's measured radiance in a given scene or data set. The same approach is used for the scenes acquired from Hyperion and ROSIS and the simulated scene. The scope of this chapter does not include the effects of atmospheric compensation, as this is not a part of the discussed level of data processing.

In the analysis of the real-world scenes, the dominating endmember of a pixel is set as ground truth. Thus it is not accounted for pixels that were not mutually exclusive i.e., consisting of more than a single endmember. This ground truth is used as a true positive for the signature sought after, whereas other pixels are marked as a true negative. The mixture relationship information of each pixel was not available from the real-world data sets [156] used in the analysis. As a result, only one endmember is associated with a single pixel.

5.4.1 Performance Metrics

This chapter investigates the effects of dimensionality reduction on signature-based target detection for hyperspectral imaging. Traditionally, SNR and ROC have been used as performance metrics in dimensionality reduction and target detection, respectively. This is a suitable approach when investigating a specific case.

The ROC is suitable to determine the optimal threshold for a specific application. However, the different ROC curves for different endmembers are not suitable when looking at the average performance for several endmembers in a given scene due to the curve not necessarily following the same trajectory for comparing cases. ROC is ill-suited when the number of cases grows large, and the analysis is given here investigates more than 2000 such cases.

SNR measures the level of restoration to the original data, which will be contaminated by noise. A lower SNR score can be due to loss of information-carrying data or removal of sensor artifacts and other undesired effects, but the exact source of the lower SNR score is often unclear. Some earlier work [144] adds noise to the scene to control that variable, but this does not solve the issue of determining the effect on the original noise.

To circumvent the limitations of SNR and ROC, it is proposed here to consider the complete chain, i.e., lossy compressing the data and performing exploitative analysis on the compressed image, to determine the performance. The analysis shows how well the combination of dimensionality reduction and target detection performs and how robust the performance is on the average case for different scenes and cases using the F_1-score and Visibility.

The analysis will determine the robustness and performance on the average case for different scenes using the F_1-score and Visibility. The target detection problem can be regarded as a binary

classification problem, so the F_1-score [145] is used as a measure of the performance of a given algorithm. The F_1-score is calculated as

$$F_1 = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} = 2 \times \frac{\text{PPV} \times \text{TPR}}{\text{PPV} + \text{TPR}}, \tag{5.27}$$

where precision or positive predictive value (PPV) is the number of true positives divided by the sum of all positives, and recall or true positive rate (TPR) is the number of true positives divided by the sum of the true positives and the false negatives. This relationship is typical in binary classification, but not as common to use in the hyperspectral remote sensing domain. The optimal F_1-score is chosen in the analysis, i.e., the highest resulting F_1-score for all possible threshold values.

Visibility [146], a measure of robustness for target detection algorithms, is also computed. A higher separability between target and background gives an easier to define detection threshold and improved suppression of undesirable false alarms, i.e., a higher level of robustness. The evaluation-metric visibility used as a measure of separability is given as

$$vis = \frac{|\bar{T}_t - \bar{T}_b|}{max(T) - min(T)}, \tag{5.28}$$

where $\bar{T}_t$ is the average response value, or probability of target for target pixels, $\bar{T}_b$ is the average response value or probability of target for non-target pixels, $max(T)$ and $min(T)$ are the maximum and minimum probability of target in the scene, normalizing the visibility value. The maximum visibility score is 1, and the lowest score is 0. By including visibility results, the scenarios where the F_1-score is high for a very narrow set of possible thresholds are avoided.

5.5 Results

Results from the real-world data are given in section 5.5.1, focusing on the performance metrics as mentioned earlier. The simulated data is given in section 5.5.2 showcases the restoration properties in terms of SNR and the degradation of computing the DR transformation matrices from a subset of data.

5.5.1 Results From Real-World Data

The subsequent plots give the averages for the F_1-score and visibility, across all available endmembers. Averages for each DR method per scene is given in Figure 5.4, 5.5, 5.6 and 5.7. Averages across all scenes are given in Figure 5.2. For all the figures the results for ACE is given in blue, CEM in red, OSP in yellow, and SAM in purple. Figure 5.3 is a more visual example of how the dimensionality reduction can affect the performance in terms of F_1-score and visibility.

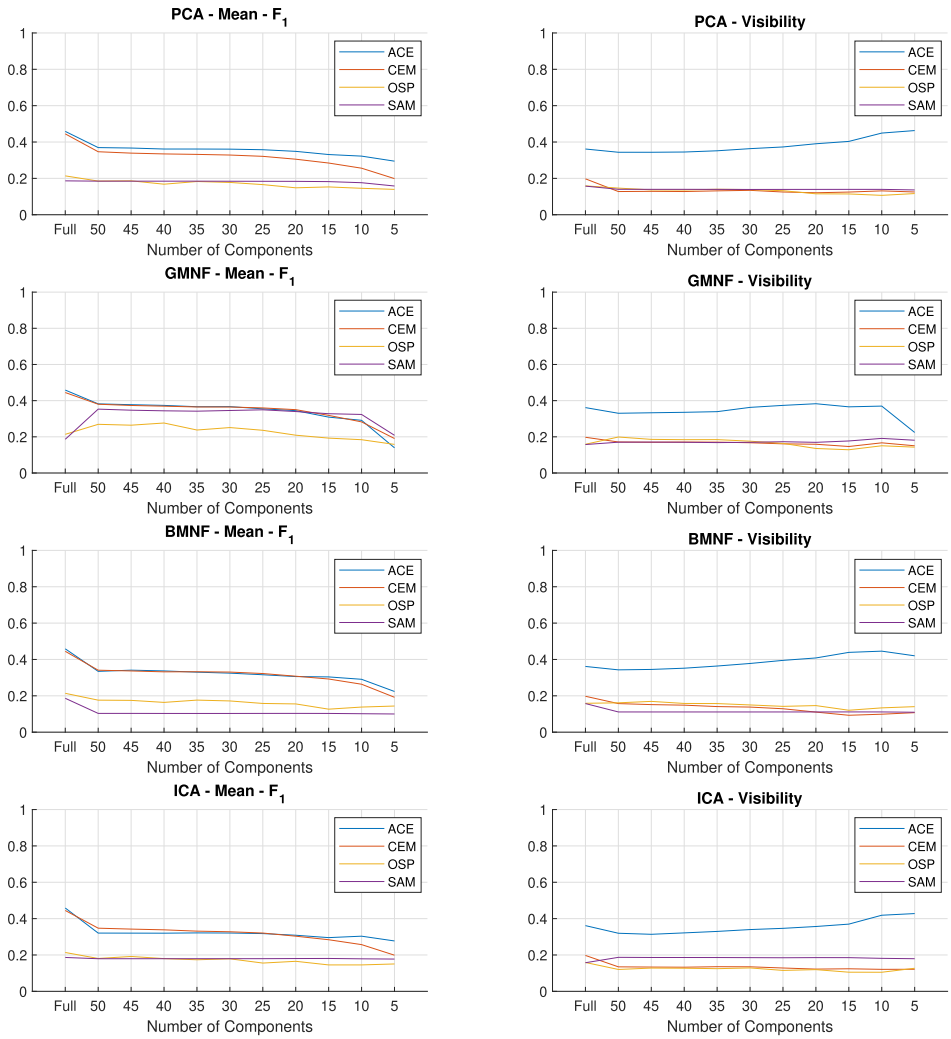


Figure 5.2.: Averages across all endmembers for all scenes for all DR transforms.

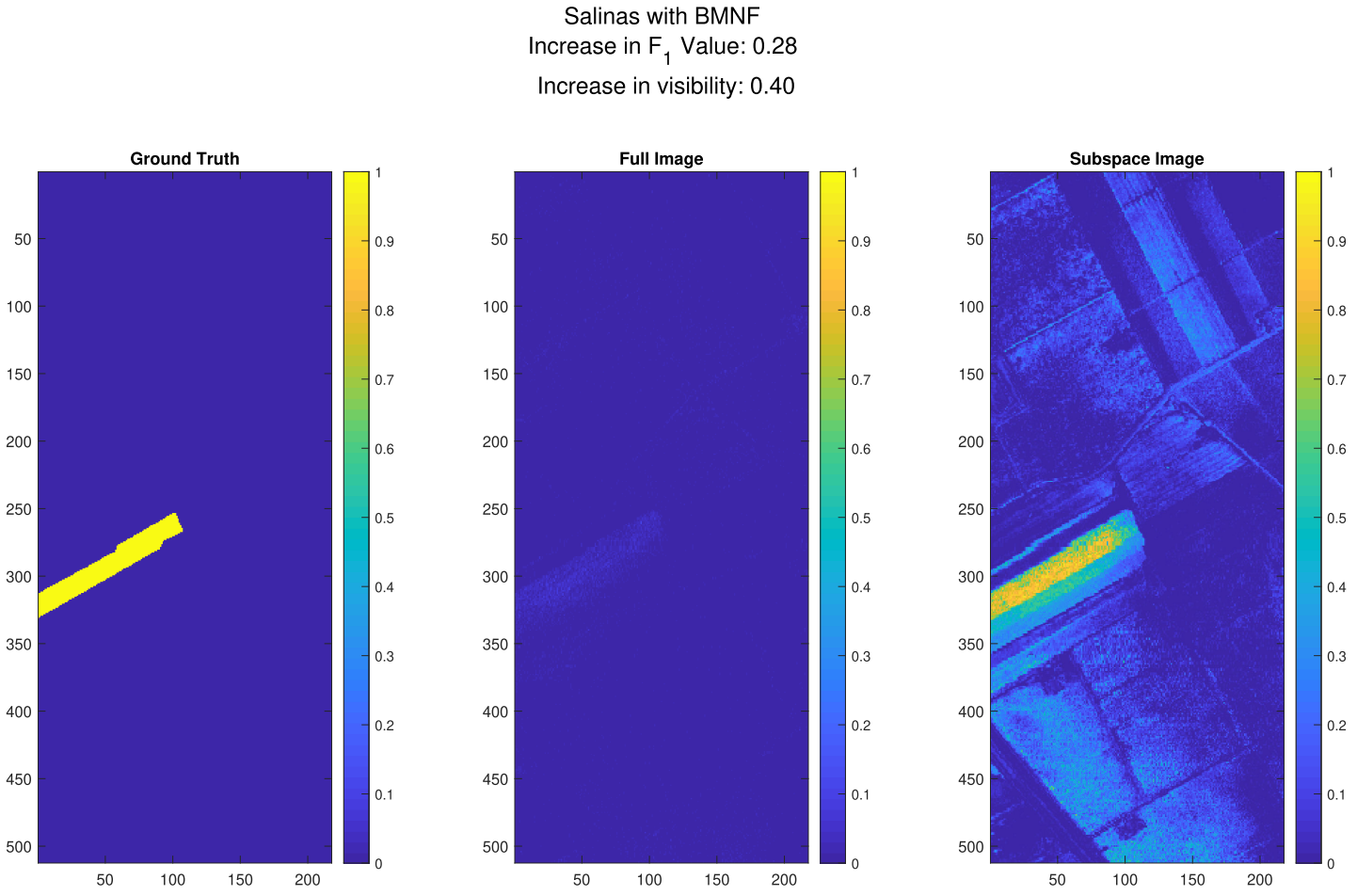


Figure 5.3.: Example of the probability intensities computed by target detection. From left to right, the probability image of the ground truth, ACE target detection on the full spectral range, and ACE target detection on the BMNF subspace using 15 components.

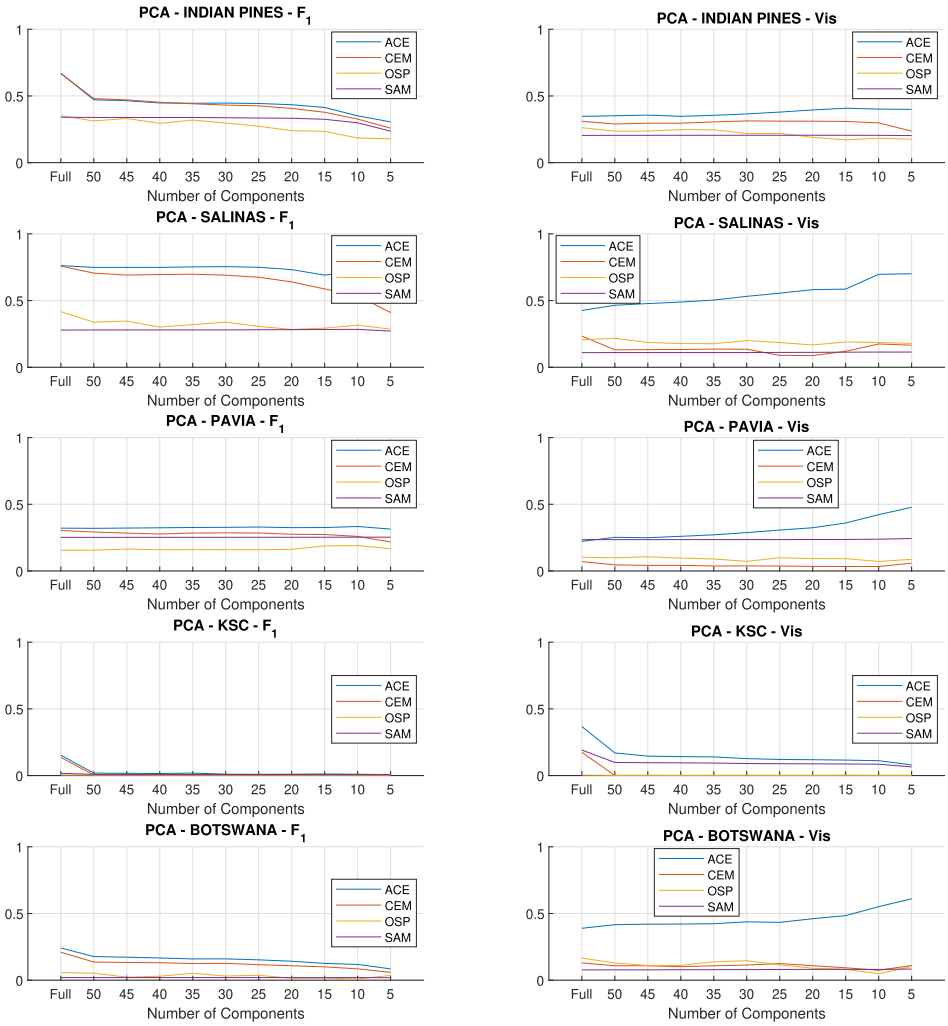


Figure 5.4.: Average performance for all endmembers per scene using the PCA transform.

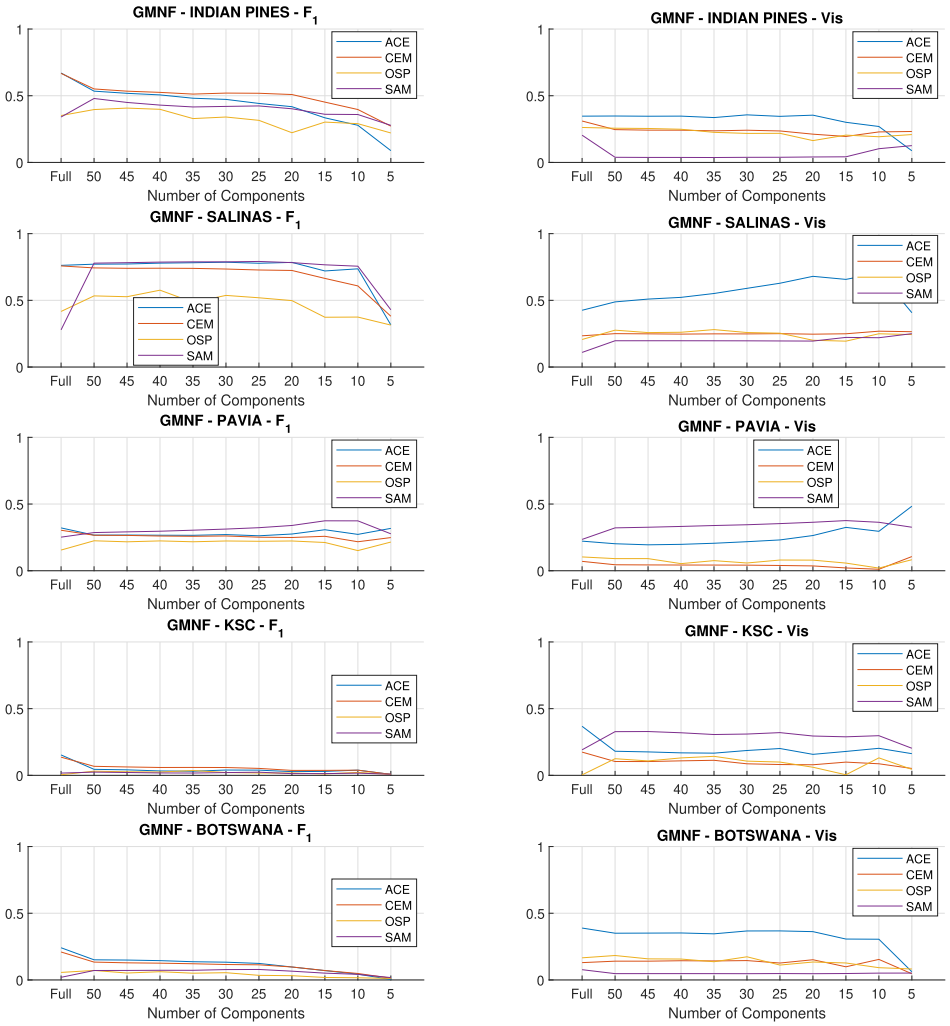


Figure 5.5.: Average performance per scene using the MNF transform with noise estimation given in equation (5.8).

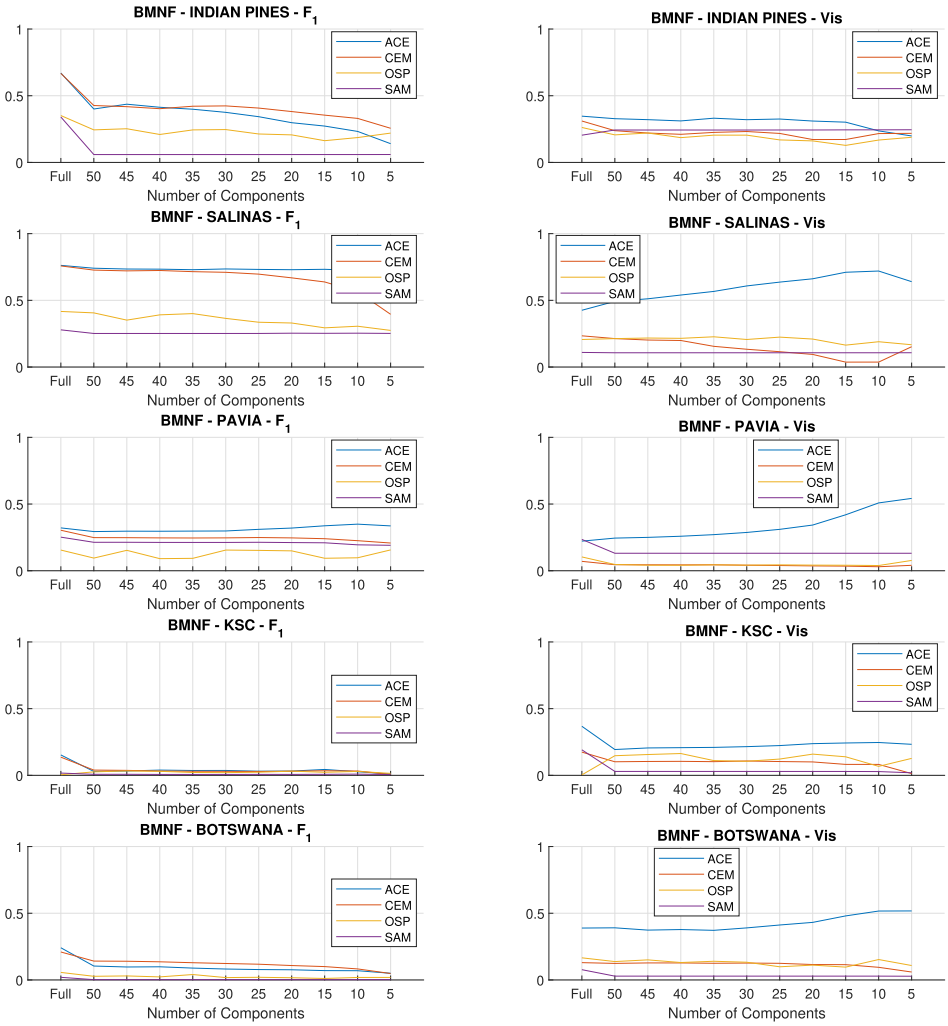


Figure 5.6.: Average performance per scene using the MNF transform with noise estimation given in equation (5.9).

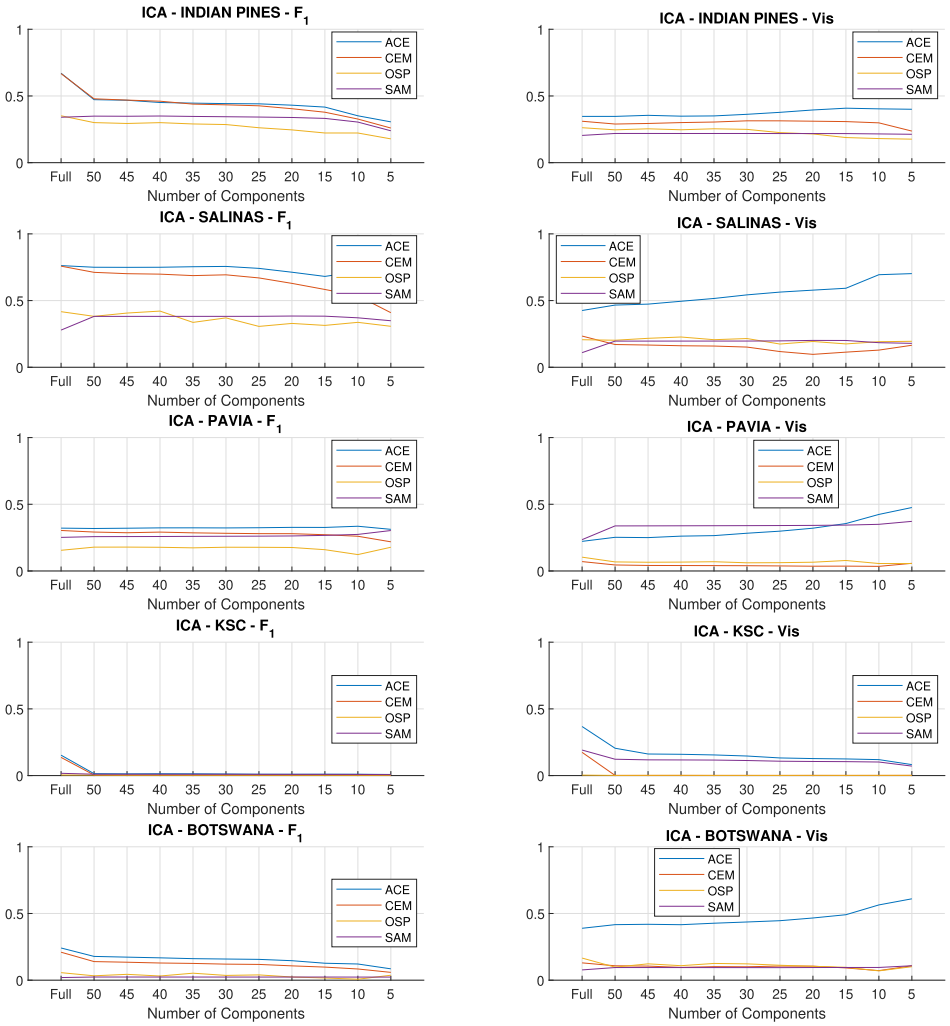


Figure 5.7.: Average performance for all endmembers per scene using the ICA transform.

5.5.2 **Results From Simulated Data**

Figure 5.8 gives the restoration performance in terms of the actual SNR of the data before noise is applied. That is, the SNR of the data compared to the spectral signatures of the synthetically generated image before adding noise and the data re-projected back to the original space via the transformation matrix, with the number of components used in the transformation matrix given along the x-axis. In addition, a comparison of the restoration performance on the full data set using components computed using all the available data and 30% randomly sampled pixels is given in Figure 5.8.

Figure 5.9 gives the noise characteristics of the noise applied in the simulated data, both in terms of standard deviation as a function of wavelength and a randomly selected noise vector.

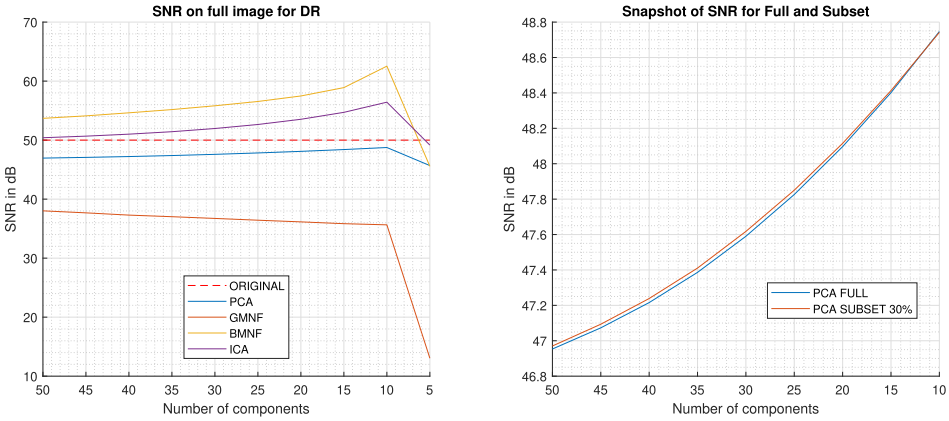


Figure 5.8.: SNR for simulated case with known noise-free signal for all DR methods.

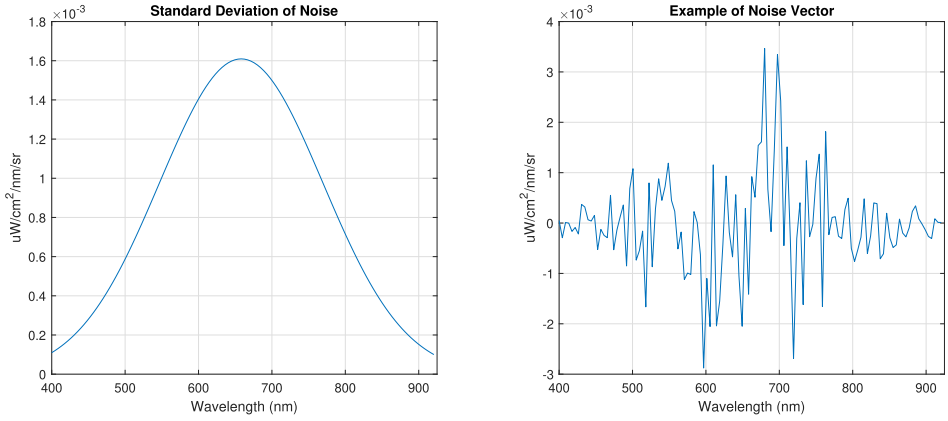


Figure 5.9.: Noise characteristics of simulated data.

5.6 Discussion and Conclusions

The target detection F_1-score on average tends to be lower when going from the total space to 50 components, as seen in figure 5.2. On average, the number of classes in the real-world data sets is 13, and when accounting for unclassified areas, the virtual dimensionality [147] i.e., the number of unique materials in the real-world data sets should exceed 10 or 15. In section 5.5.1 the effect of this can be seen in the decreased F_1-score when using fewer components than the assumed virtual dimensionality. Reducing the number of components further from 50 does not seem to affect the F_1-score significantly up until the assumed virtual dimensionality is reached.

As seen in figure 5.2 the data can be analyzed in the subspace without significant loss in terms of target detection visibility on average, and in some cases, the visibility tends to increase with the reduced number of components. The overall visibility performance found for the ACE detector corresponds well with the favorable results found in other publications comparing detection algorithms. [146, 152, 154, 163, 164]

Both GMNF and BMNF perform well on the Salinas scene, where the assumption of homogeneous areas are well met, as seen in figure 5.5 and 5.6. Compared to their respective results for the heterogeneous KSC scene, the performance decreases. An example of a good case for the BMNF transform is given in figure 5.3. This indicates that a more suitable noise model can improve the detection rate and visibility of a given target or application. An inaccurate noise model for the data can degrade the performance. However, a non-ideal noise estimation does not significantly degrade the overall performance. From the average statistics per scene, it is evident that when the assumptions made to create the DR model are erroneous, a more simplistic approach e.g., PCA, seen in figure 5.4, is just as suitable if not more suitable than the more complex dimensionality reduction methods in terms of target detection in the subspace. The more complex JADE ICA method, seen in figure 5.7, did not yield a good performance when compared to the aforementioned PCA, or MNF. However, as seen in figure 5.9 the ICA and BMNF transform can restore the original signal in the synthetically simulated data better than PCA. On average ICA and BMNF is able to remove noise when the number of components used are more than or equal to 10, the virtual dimensionality that the number of classes used in the synthetization would suggest.

The simplicity of implementation should also be taken into consideration. For the dimensionality reduction methods surveyed, the JADE ICA transform is by far the most complex and computationally demanding, while the PCA is the simplest. The MNF could be regarded as a double PCA, a whitening step, and derivation of components. The most computationally demanding and complex of the surveyed target detection algorithms is the OSP detector, which gives relatively poor results. The ACE and CEM detector are of similar complexity and give similar results, but the ACE detector tends to be more robust in terms of visibility. Lastly, the SAM detector is the simplest and gives encouraging results when combined with the GMNF transform.

The reduced performance seen in figure 5.9, shows that there will be a minor change in restoration performance when computing the DR transformation matrix using only 30 percent of the available data as opposed to all the available data in terms of SNR. Thus, a transformation matrix computed with only 30 percent of the data should still yield a good analysis of the scene, both in terms of restoration to an original noise-reduced scene and target detection performance in the subspace. As computing the DR transformation matrix can be very computationally expensive, this is a promising result showing data is not necessarily much to gain from using all the available data. This result has two potential pathways for remote sensing smallsats; if the transformation matrix is computed on-board, it is not needed to account for all the available data, and the performance decrease from using a pre-computed transformation matrix can in some cases be negligible i.e., if the a priori assumptions is a good representation of the scene.

All the discussed spectral dimensionality reduction methods give a compression ratio that can be easily computed, as shown in Table 5.2. The spectral dimensionality reduction can be conducted as a preliminary compression step prior to lossless compression methods e.g., JPEG2000, CCSDS123 [79]. The information lost in the dimensionality reduction will consist of noise in the statistical sense, as seen in figure 5.8. Moreover, later lossless compression stages can ensure a lower total data volume for the resulting data cube.

If the DR transformation matrix is computed based on the expected endmembers of a scene rather than the whole scene, this could save both time and power for a given remote sensing smallsat system. The change in performance seen in this analysis when using only 30 percent of the scene to compute the DR transformation matrix is negligible. How well the DR transformation would project a given scene will depend on the specific application, but the results given here indicate that the change in performance on average could, in many applications, be acceptable, and in some cases, beneficial.

5.6.1 Future Work

An adaptive approach to statistically estimate an optimal number of components used for dimensionality reduction [4, 147] could be beneficial to investigate. With such an approach, the number of components selected does not need to be dependent on a satellite operators' input but rather the statistical relationships in the data. It is currently not investigated if the state-of-the-art methods used to estimate an optimal number of components i.e., virtual dimensionality for dimensionality reduction, are optimal for the performance metrics given here.

Here, the noise model for the MNF DR methods is computed on a per-pixel basis, thus having the underlying assumption that all pixels will exhibit comparable noise characteristics. For many electro-optical systems, this can be an erroneous assumption, such as for the pushbroom HSI used in HYPSO. A better noise model, based on a sensor model and accounting for the different sensitivities of the focal plane array, as opposed to a per scene model or per pixel model, could

make the MNF DR transformation matrix further suppress undesired noise or other artifacts from the data.

It is possible to conduct analysis or data exploration in the subspace; how an application-specific analysis is affected by being performed in the subspace should be investigated. Performing an analysis in the subspace could save the number of operations needed, and thus reducing the computational strain, potentially increasing the throughput as an edge computing agent, such as smallsats for Remote Sensing.

Chapter 6

Compression with Residual Analysis for Hyperspectral Remote Sensing

I find the lure of the unknown irresistible.

Sylvia Earle

The demand for high-resolution hyperspectral images from remote sensing is often in conflict with the available transmission bandwidth or storage capabilities. Here, we propose a combination of well-established methods for spectral and spatial image encoding using JPEG2000 and demonstrate their synergistic effects. The proposed pipeline consists of dimensionality reduction, wavelet transformation, residual analysis, and source coding. The dimensionality reduction is based on a particular version of Principal Component Analysis developed for Big Data. It splits the high-dimensional data stream into systematic variation patterns and noise variations in the spectral domain. These variation types are compressed separately, allowing flexible compression, ranging from highly informative reconstruction quality at very low bit-rate, to near lossless reconstruction quality at a much higher bit-rate, when non-systematic and systematic errors are also compressed. The compression performance is evaluated using publicly available hyperspectral data from AVIRIS andHICOwith Signal Noise Ratio and Spectral Angle Map as quality metrics. A comparison based on the results of PCA-based transformation is given. The proposed pipeline expands upon transformation based compression with regards to the discovery of new, unexpected variation patterns. An increase of Signal to Noise Ratio of up to 2 dB at 77% of the computational time is achieved with respect to the compared method. This shows potential for lossy compression performance for hyperspectral images, with controllable loss handling and near-lossless reconstruction.

6.1 Introduction

Hyperspectral imaging (HSI) makes it possible to sample parts of the light spectrum with a higher density and in other wavelength regions than with traditional imaging systems, such as RGB

cameras. This is important for many applications, e.g., satellite monitoring of environmental challenges. This high spectral resolution creates very high data volumes. In some applications, a high spectral resolution can be helpful but is characterized by high redundancy. Therefore, mitigation actions to reduce the data volume can be taken when the cost of data transfer or timeliness of the information is important. Typically, lossless compression methods require a bit rate that is higher than lossy compression methods in terms of bits per pixel per band (bpppb) [165–167], but retains the capability to restore the data fully. For some HSI applications, the timeliness of the information requires even higher compression rates to be relevant for transmission. An approach that allows control of the bit rate so that the most relevant and reliable image information is given higher priority than what statically can be considered random noise is proposed. This proposed method decreases the data transmission volume at the cost of higher computational complexity. A residual analysis to inspect the data lost in the compression is presented to help control the loss.

HSI cubes have both spectral and spatial extent that are highly inter-correlated. Compression algorithms designed for HSIs take advantage of these redundancies to reduce the data volume. The compression generally consists of using a transform-based or prediction-based decorrelator for the spectral and spatial dimensions, followed by a quantization stage and an entropy coder [167–169]. With the high data rate of current HSI sensors, it is attractive to use near-lossless, and lossy compression techniques [168, 169]. Furthermore, transform-based approaches have been shown to perform well for lossy compression [168], when compared to prediction-based methods. Transform-based lossy compression approaches, in particular the Principal Component Analysis (PCA) coupled with the JPEG2000, for decorrelating the spectral and spatial information respectively, have yielded favorable results in terms of rate-distortion and information preservation [168, 170–174]. Dimensionality reduction by PCA has proven useful for hyperspectral data representations [4, 168, 173, 175] for further analysis as well. While PCA gives optimal decorrelation features, it is also characterized by a high computational cost and intensive memory requirements [168, 173, 176–178]. Moreover, it requires a choice of how many PCA components to use in the data-driven image modelling. This can make PCA approaches less suitable for applications with latency, power, or memory concerns, unless mitigation strategies are used [172–174].

The JPEG2000 standard Part-1 and Part-2 is a 2D image coder, whilst Part-10 is designed for three-dimensional coding [179]. However, JPEG2000 Part-10 is created for isotropic data sets that have a spatial interpretation in all dimensions and is thus not well-suited for data-types with both spatial and spectral content [179]. For a PCA and JPEG2000 based HSI compression the JPEG2000 encoding can be computationally demanding [176].

The results from the spectral transform by PCA, also known as dimensionality reduction, are here referred to as scores. This is a linear combination of spectral variables related to the original data by a transformation matrix, referred to as loadings [4, 84, 143, 180]. The spectral loadings are costly to compute, but needed for transforming the original data to scores [143, 168, 172, 173, 176, 180]. By using a pre-computed set of spectral loadings, the fidelity of the scores is often retained throughout

the transformation of new but similar scenes [84, 143, 181]. This is not necessarily true if new or unexpected phenomena are captured during the acquisition. The PCA reduced data representation has shown to retain sufficient information for HSIs for further exploration in its compressed form [4, 80, 84, 143, 175, 182], and can still be interpreted spatially.

Several studies have combined PCA and JPEG2000 to achieve high compression performance for lossless [165, 174, 178, 179] as well as lossy [166–168, 170–172, 176, 179, 183] compression of HSIs. These two decorrelation steps are throughout referred to as stage one of the proposed compression method and are used as the baseline for comparison. However, to the best of the authors' knowledge, previous work has not addressed the model discrepancies that occur when using transformation-based encoding as proposed here [167]. Lossy compression approaches typically have low average distortion, but are not able to make any assurances about the fidelity of any given set of samples [169]. Without any analysis of residuals, new or poorly modeled phenomena are still not accounted for.

We define residuals as the difference between the original data and the lossy PCA transformed data. Analyzing the residuals is here referred to as stage two of the compression. These residuals are analyzed to find systematic errors indicative of unmodeled or poorly modeled phenomena. When using transformation-based encoding, the significant residuals can indicate systematic errors in the transformation. Analysis of residuals is proposed to determine which are significant, and these large residuals are given priority. This online analysis of lossy compression can provide information about the potential new trends that can later be incorporated into updated pre-computed loadings [84, 184], e.g. online PCA. This is not shown here.

The second part of the proposed processing pipeline expands on transformation-based compression approaches using PCA by computing residuals to provide information on data loss during the compression and an option to retrieve that information wholly or in parts. Using residuals can result in lossless reconstruction [165, 178]. However, it is intended as a reassurance measure to know that no significant unmodeled phenomena were lost due to lossy compression. To have control of the bit rate, the residuals are compressed separately, demonstrating how the proposed approach can provide greater confidence in the lossy compressed data at a predetermined bit rate. Low bit rate cubes from stage one combined with high bit rate analyzed residuals achieve a reduced total computational time at similar or higher quality, in terms of SNR, compared to the results achieved by high bit rate cubes from stage one.

The chapter is organized as follows; Section 6.2 details the theory used, Section 6.3 describes the simulation and its implementation with results and discussion presented in Section 6.4. The conclusions are found in Section 6.5.

6.2 Background

HSI cubes have two spatial dimensions and one spectral dimension [4]. Here, a pixel or feature refers to a point in the spatial dimensions that contains the entire spectrum, forming a vector. The

data matrix for a given data set is given in eq. (5.3). The two spatial dimensions are reshaped into one dimension by concatenating rows in the original spatial 2D coordinate system, yielding N column vectors for the N pixels. Column vector x_i is the spectrum of pixel i.

6.2.1 Spectral Decorrelation by Dimensionality Modeling and Reduction

HSIs typically have high spectral dimensionality, and the captured electromagnetic spectrum that is sampled at a high resolution can carry redundant information [143, 147, 174, 175, 182]. Dimensionality reduction, in a sense a special case of band selection, attempts to reduce the noise in the data set by eliminating redundant information and reducing computational costs while preserving the significant spectral information [185]. There are many strategies for reducing the dimensionality [4, 147, 173, 174], but here only the PCA is considered. Analyzing this redundancy makes it possible to find an appropriate subspace to represent the original data within. Spectral decorrelation is used here for reducing the dimension of a spectrum [4, 84, 147, 173, 175] both for compression and to limit computational cost of further processing.

The PCA algorithm gives optimal decorrelation features when assuming Gaussian noise [4, 168], and is thus used for decorrelation. However, in this chapter, the combination of PCA and JPEG2000 score compression is expected to reduce the detrimental effect of too many principal components. The reason is that since the sequence of PCA loading vectors are scaled to be orthonormal, the corresponding row of score images are expected to have about the same noise level and hence progressively lower SNR. Consequently, the variation that *survives* the JPEG2000 score compression will be progressively smaller. In the literature referenced in the introduction, the ones that use the JPEG2000 image compression standard mainly use the multi-component transform without allowing any loss in the image encoding performed by JPEG2000.

Dimensionality reduction enables the usage of algorithms unfit for high dimensional data, as it retains related information even in its compressed form [80, 84, 143, 175, 180]. The subspace also has spatial interpretability in its dimensionally reduced state by this property. Selecting an optimal number of dimensions for the subspace can be complicated. By selecting fewer dimensions, there is a risk of losing potentially important information, whereas by selecting many dimensions, the model starts representing what can be considered as noise [4, 143, 147, 175, 180]. Thus, selecting dimensions to use will ultimately depend on the application.

In this chapter, combinations of components have been tested with various JPEG2000 encoding parameters for a set of desired bit rates. Analysis of the residuals can also aid in determining the appropriate number of components [84], as well as finding if the model does not account for any new trends or anomalous phenomena.

Noise Characterization

HSI sensors are susceptible to noise [4]. By comparing the artifacts from the compression algorithm with the noise characteristics found in the original data set [166], it is possible to gain a better

understanding of the distortion from lossy compression. An intermediate step of the subspace identification algorithm presented in [147] and in chapter 5 is used to estimate the noise in the data sets. This intermediate step is given below. The noise for a given pixel is estimated by inverse matrix interpolation. Noise vectors $\mathbf{X}_n$, with L as the number of spectral bands, are estimated using eq. (5.9) [4]. Where the estimated signal matrix $\mathbf{X}_s = \mathbf{X}_o - \mathbf{X}_n$ is compared with $\mathbf{X}_o$ to estimate the noise characteristics.

Principal Component Analysis

Principal Component Analysis (PCA) is a common method that effectively decorrelate spectra by utilizing spectral redundancy [4, 173, 175, 180, 186]. Within PCA, the principal components are ordered according to decreasing eigenvalue. The dimensionality can be reduced by only using a selected number of principal components with the largest magnitudes for the transformation. The eigenvalues σ^2 and principal components can be computed as in eq. (5.4). Where Σ_o is the sample covariance matrix computed from HSI data matrix $\mathbf{X}_o$, $\mathbf{I}$ is an identity matrix with L dimensions, and $\mathbf{m}$ is the mean spectrum. The loadings matrix with unit eigenvectors $\mathbf{V}_{PCA}$ is ordered according to the magnitude of the eigenvalues, such that

$$\Sigma_o \mathbf{V}_{PCA} = \mathbf{V}_{PCA}\mathbf{D}, \tag{6.1}$$

where $\mathbf{D}$ is a diagonal matrix consisting of decreasing eigenvalues, and $\mathbf{V}_{PCA}$ is a matrix consisting of the corresponding unit eigenvectors that fulfill $\Sigma_o \mathbf{v}_j = \sigma_j^2 \mathbf{v}_j$. Dimensionality reduction is performed by projecting onto a subset of the eigenvectors related to the largest eigenvalues in $\mathbf{V}_{PCA}$, in the following way

$$\mathbf{Z}_{PCA} = \mathbf{V}_{PCA}^T \mathbf{X}_o, \tag{6.2}$$

where $\mathbf{Z}_{PCA}$ are the resulting scores for the selected number of components. The first components carry more information, as illustrated in figure 6.1, where the components are scaled by their eigenvalue. Figure 6.2 shows the resulting scores from the spectral decorrelation. The spatial interpretability in the subspace is illustrated in figure 6.2b.

6.2.2 Spatial Compression by Wavelet Transform

Natural images constitute only a tiny fraction of the possible values an image cube can take [180]. The redundant spatial information can be identified and removed with a suitable transform basis after dimensionality reduction. Two-dimensional wavelet transform has been shown to give a sparse representation of natural spatial information [171, 180, 187]. Wavelets are characterized by a multi-resolution decomposition that accommodates differences in space and frequency. This is a crucial feature when decomposing signals from multi-scale processes such as spatial images [180].

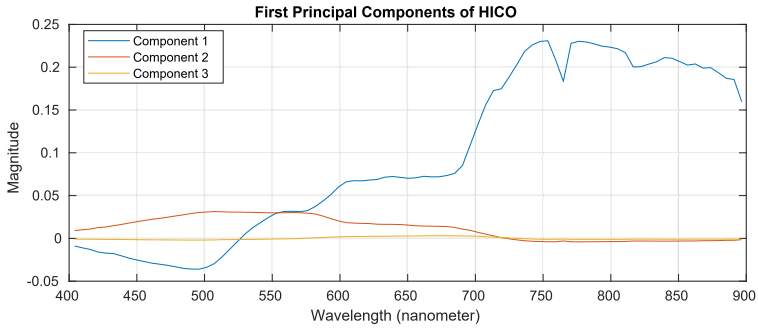
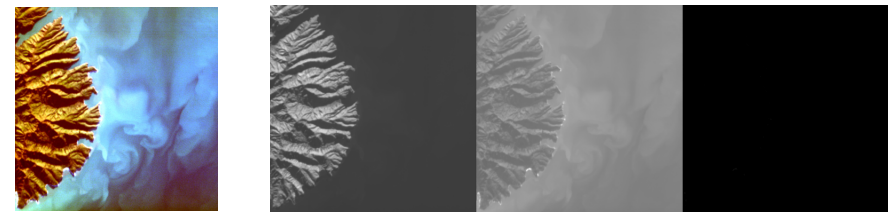


Figure 6.1.: The three first PCA components in theHICOChristchurch scene from PCA, multiplied by the corresponding eigenvalue.



(a) **(b)**

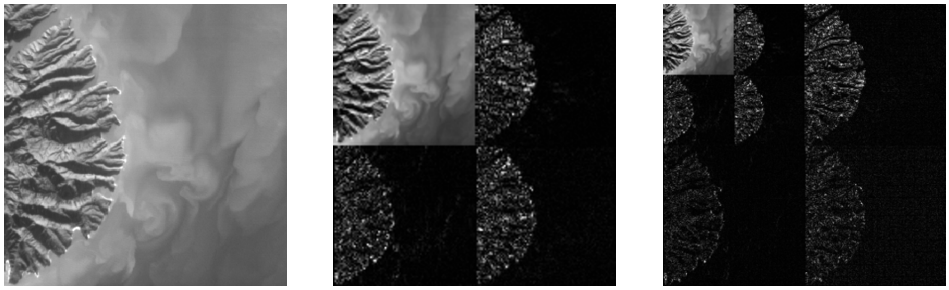
Figure 6.2.: RGB Reconstruction of theHICOChristchurch scene 6.2a, accompanied with the three first score plots 6.2b of the same spatial scene, in increasing order from left to right.

Wavelet transform handles transient signals, i.e., signals with sudden changes. From a signal perspective, realistic spatial images consist of sudden changes with different characteristics at different scales, and can be efficiently described in the wavelet domain. Signals that are smooth and periodic are generally compressed more efficiently by other methods, such as the Fourier Transform [180, 188]. More in-depth discussion of wavelets is presented in [180, 188, 189].

The discrete Wavelet Transform (DWT) transforms spatial images into the wavelet domain to perform compression by removing low-energy components. For image transform by wavelets in two dimensions, here denoted as x and y, four different functions are needed. Firstly a two-dimensional scaling function $\phi(x, y)$. Secondly, three two-dimensional wavelets for details in different direction; $\psi^H(x, y)$ for horizontal details, $\psi^V(x, y)$ for vertical details, and $\psi^D(x, y)$ for diagonal details. Down-sampling is used for the scaling function. How the different details will be captured depends on the wavelet basis used.

Wavelet decompositions can be viewed as performing low-pass filtering by down-sampling and high-pass filtering to get the details, in a way that satisfies a given set of constraints [188]. The process is then repeated on the down-sampled data.

The sparsity that the transform provides is presented in figure 6.3. The first downsampling can be seen in the top left corner of figure 6.3b. The horizontal, vertical, and diagonal details are given in their respective direction. Lastly, wavelets can be extended further to three dimensions or beyond [165, 179], but an appropriate wavelet basis for both spectral and spatial data needs to be determined.



(a) Original **(b)** Two decompositions **(c)** Three decompositions

Figure 6.3.: Photo of Christchurch, New Zealand with a decomposition level progression from original, to two and then three decompositions.

JPEG2000 on Hyperspectral Image Cubes

The image compression standard and coding system JPEG2000, as made available by the reference software found in [190], has been used in this chapter. JPEG2000 uses wavelets to compress images. This coding standard is developed for natural 2D images, and isotropic 3D images [179], i.e. not explicitly developed for hyperspectral image cubes. It does support multi-component transform, i.e., a transformation of images with multiple components by means such as a PCA transform.

Spectral decorrelation is used to increase the compression performance of the JPEG2000 standard when applied to HSIs [170, 171]. The resulting scores from spectral decorrelation are sorted to have spatial interpretability, utilizing that scores retain information in its compressed state, see Figure 6.2b.

The artifacts from JPEG2000 compression can appear as blur and rings near edges in the image with aggressive compression [188]. These artifacts are also present when compressing HSIs via the JPEG2000 encoder. JPEG2000 provides both lossless and lossy compression in a single compression architecture, and here only the lossy compression is used, as opposed to most other implementations using JPEG2000 on hyperspectral imagery.

6.3 Methods

The processing pipeline proposed in this chapter consists of two main stages, co-variation analysis, and residual analysis, and is visualized in Fig 6.4. The code is made available at [191].

The first stage is the baseline HSI compression scheme using JPEG2000, with the possible addition of dimensionality reduction by projection from known loadings from PCA and 2D-conversion, i.e. concatenation at each wavelength into a two-dimensional spatial image. This 2D conversion reduces the assumptions made by the encoding for a three-dimensional HSI structure, by having multiple natural 2D images side-by-side. Four different modes for compressing the hyperspectral images are compared for the first stage. They are abbreviated as follows:

1. J2K3D - The image cube compressed using multi-component JPEG2000 encoding.
2. PCA3D - The PCA spectrally decorrelated score cube compressed using multi-component JPEG2000 encoding.
3. J2K2D - The image cube, concatenated at each wavelength into a two-dimensional image, compressed using two-dimensional JPEG2000.
4. PCA2D - The PCA spectrally decorrelated score cube, concatenated at each component into a two-dimensional image, compressed using two-dimensional JPEG2000.

The second stage consists of performing the residual analysis and utilization of the results to infer the data loss from the initial stage, i.e., using only dimensionality reduction by PCA. In this stage, the residuals are processed without 2D-conversion due to potential spatial sparsity after residual analysis. After residual analysis as described in Section 6.3.2, the residuals are decorrelated using PCA. The bit rate is controlled by encoding the decorrelated residuals with the JPEG2000 encoding scheme. An attempt at illustrating this is given in Fig 6.4.

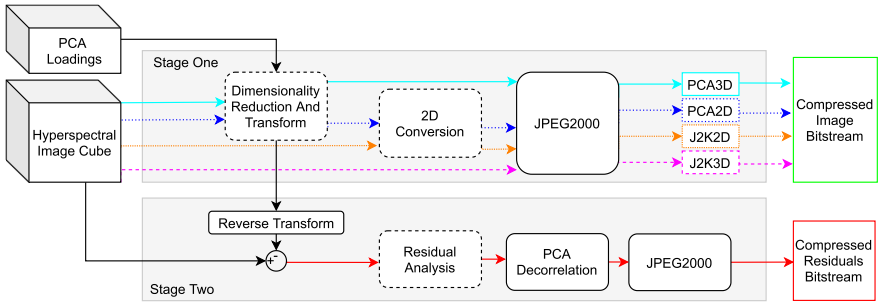


Figure 6.4.: The proposed two stage processing pipeline.

6.3.1 Data Sets

The data sets from AVIRIS of Cuprite and Moffett scenes, andHICOof Christchurch scene representing land, urban, and ocean scenes, respectively, are used for training and testing. This is, unfortunately, a minimal data set for validating results. However, using different training validation data sets is intended to improve the validity of the proposed method on new images and varying scenes. By using varying scenes representing a broad set of topics for typical Earth observation

applications, we hope to improve the validity of the results further when relying on these limited data sets.

The data sets are divided into a set for training the PCA model, and a testing set. In Fig 6.5 and 6.6 the areas marked with a black square in the bottom right corners are used for testing, and the remainder is used for training.

The AVIRIS data sets are used for comparison with [170, 171, 179]. Here, the flight log ID and RGB reconstructions are included to compare results easily. From the datasets, a total of 36 bands were discarded due to water absorption as well as lousy sensor characteristics [171]. The removed bands are [1-2, 104-113, 148-167, 221-224]. The remaining frequencies are found in Figure 6.7.

TheHICOdata sets are included to explore if the approach works for spatially homogeneous natural targets, as well as for ocean color applications. For the HSI cube of Christchurch, New Zealand, a total of 41 bands are discarded due to lousy sensor characteristics [192]. The removed bands are [1-9, 97-128], corresponding to 352 to 398 and 902 to 1080 nm. The data sets are cropped to 320 by 320 pixels spatially. This fixed size makes the results from the different scenarios easier to compare and present invariably. Unfortunately, this makes it more difficult for other researchers to compare their proposed methods with the given results. The training and testing sets are selected as shown in Figure 6.5 and 6.6, and information about all the data sets can be found in Table 6.1. BQ is a measure of *naturalness*, and details can be found in Section 6.3.3 [193].

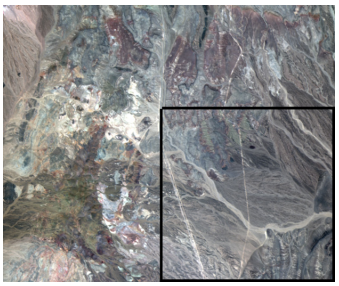
These data sets represent calibrated hyperspectral remote sensing data. The calibration coefficients used to correct the hyperspectral image will create artifacts that can be exploited as these corrections will create repeatable patterns in the data, i.e., redundancy. In the past, these types of calibrated data sets were exploitable in terms of compression performance [194]. This exploitability is not used here, not in the baseline or proposed method.

Table 6.1.: Data Sets Used

Name	Sensor	Flight Log ID	Spatial Resolution	Number of Spectral Bands	BQ	Source
Cuprite	AVIRIS	f080611t01p00r06	320x320	188	44.29	[158, 195]
Moffett	AVIRIS	f080611t01p00r07	320x320	188	41.06	[158, 195]
Christchurch	HICO	H2011167024711	320x320	87	36.13	[192, 196]

Quantization of Spectrally Encoded Data and JPEG2000

Quantization maps values to predefined bins. This enables any data to be represented using fewer bits at the cost of lost precision. Different quantization methods will trade either data fidelity or computational time. In our implementation, after dimensionality reduction, the scores are floating-point numbers [165], here represented as 64-bit doubles. As Openjpeg [190] only supports 16-bit representations, quantization is one viable option to reduce the bit depth. Here, data is quantized by scaling from 64-bit doubles to a 16-bit unsigned integer representation without further optimization. These quantization errors are included in the residuals to mitigate their associated distortion. This quantization transform is uniform, and the rate-distortion weights are not used to



(a) Cuprite with the cropped testing cube in the black square, representing a land target.

(b) Moffett with the cropped testing cube in the black square, representing an urban environment.

Figure 6.5.: RGB reconstructions of AVIRIS datasets with testing sets in the black squares.

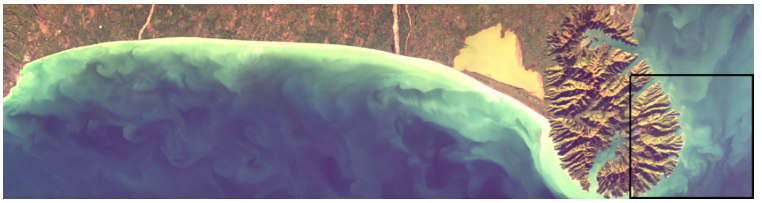


Figure 6.6.: RGB reconstructions ofHICOdataset with testing set in the black square, representing an ocean color application of hyperspectral imagery.

scale coefficients to an optimal dynamic range. If this is performed, it is expected to improve the performance at a higher computational complexity. A more optimal quantization strategy in this part of the compression should also yield better performance at the cost of higher computational complexity, but this is not further investigated here.

The Lossy JPEG2000 compression is irreversible as it also introduces quantization noise in the rounding step. After the wavelet transform, the coefficients are scalar-quantized to reduce the number of bits for representation, and a larger quantization step would yield a larger compression ratio at the cost of *data fidelity*. This cost is quantified by the metrics given in Section 6.3.3. Here, 64 by 64 blocks of spatial pixels are transformed into the wavelet domain.

The transformed blocks are sent through Embedded Block Coding with Optimal Truncation (EBCOT). This ensures the encoding of bits of quantized coefficients of a block. These encoded bits are then put through the binary MQ-coder, a context-driven binary arithmetic coder [188].

6.3.2 Computation and Analysis of Residuals

The residuals, i.e., the discrepancy between the original and decompressed image cube, are further processed in the second stage of the compression approach, illustrated in as a red arrow in Figure 6.4. This second stage analyzes the residuals, identifies significant ones, and then decorrelates and compresses them.

The residuals are decorrelated using PCA. The resulting score matrix transforming the residuals is then compressed using JPEG2000 to retain a predetermined bit rate and effective encoding. This second decorrelation using PCA of the residuals is done so that it can be more effectively encoded using JPEG2000. The resulting decorrelated residuals are encoded in their 2D matrix form and interpreted as a regular image by the JPEG2000 encoding scheme. The residuals are defined in two ways; non-zero residuals and large residuals, which exceed one standard deviation in a Gaussian normal distribution. They are denoted with *ALL* for non-zero residuals and *1STDV* for the residuals that exceed one standard deviation from the zero mean assuming a Gaussian distribution, respectively.

The sample mean and standard deviation are computed at each spectral band for the residuals. When any residual related to a spectral band exceeds the magnitude of the associated standard deviation from the mean, it is considered a significant residual. If any of the residuals at a spectral band for a pixel has a greater magnitude than the standard deviation, it is kept.

With the residuals as a separate compression outcome, it is possible to determine which pixels are adversely affected by the compression, and transmit their residual. In [165], a similar processing scheme was demonstrated to achieve lossless compression at a bit rate of 5.52 and 5.73 for a similar Cuprite and Moffett scene, respectively, from the AVIRIS sensor. This processing scheme used 24 bits to encode the scores. Compared to the specific data sets used here, a lossless bit rate of 8.51 for the Cuprite scene and 9.65 for the Moffett scene was achieved by compressing with lossless JPEG2000.

6.3.3 Metrics

Different metrics are used to measure the effects of compression. To enable better comparison with similar works, the measure *bits per pixel per band* (bpppb) is used to indicate the compression performance. The restoration to the original data cube is represented by the signal-to-noise ratio (SNR). The SNR is given as the log ratio of signal variance to mean squared error [166,170,171,179]. Additionally, a measure of the average spectral angle mapper (SAM), the mean angle of the corresponding spectra between the original and the reconstructed cube, is given. Lastly, the BRISQUE (BQ) score is also given for image quality assessment without using a reference image. The BQ-score is used to indicate how much like a natural image an HSI is per wavelength and implies whether or not noise is removed. This metric is used in place of target detection or anomaly detection as a performance metric, as these methods will be dependent on well-labeled data, and the performance of the chosen algorithm used for benchmarking [80, 170, 172].

Signal-to-Noise Ratio (SNR)

The SNR is a common metric for indicating the total loss when decompressing a signal or an image [188]. A high SNR is desirable. In [172], for multiclass classification and spectral unmixing, the SNR metric was shown to be positively correlated with classification performance across many

decorrelations strategies. The SNR is computed as decibel, the log ratio of signal variance to mean squared error, as

$$SNR = 10\log_{10}\left(\frac{\sum_{k=1}^{K}(H_o(k)-\mu(H_o))^2}{\sum_{k=1}^{K}(H_o(k)-H_r(k))^2}\right), \tag{6.3}$$

where H_o and H_r are the original and decompressed HSI cube with K elements, and $\mu(\cdot)$ is the mean function.

Spectral Angle Mapper (SAM)

This is a measure of the angle between two vectors containing spectra, and can be used for signature-matched target detection [4]. The angle indicates the similarity between two spectra, and thus a low angle is desirable. Here the spectra from the original HSI cube are compared with the corresponding spectra from the decompressed cube, indicating what level of distortion each spectrum has succumbed to by lossy compression. The angle discrepancy is given in degrees. With $\mathbf{t}_i$ and $\mathbf{r}_i$ as the original and decompressed spectra vectors the SAM is computed as

$$SAM = -\cos^{-1}\left(\frac{\mathbf{t}_i^T\mathbf{r}_i}{\sqrt{(\mathbf{t}_i^T\mathbf{t}_i)(\mathbf{r}_i^T\mathbf{r}_i)}}\right). \tag{6.4}$$

BRISQUE (BQ)

The *Blind/Referenceless Image Spatial Quality Evaluator* (BRISQUE) metric is a blind image quality assessment that evaluates how natural a scene looks in the spatial domain, here abbreviated as the BQ-score or simply BQ. The BQ-score uses scene statistics of locally normalized coefficients for luminance to quantify the possible losses of *naturalness* in the image due to distortions [193]. It has been shown that BRISQUE is better in terms of correlation with human perception than the full-reference peak signal-to-noise ratio and the structural similarity index in determining distortion. It is also highly competitive to all present-day distortion-generic when no reference is given with natural images [193].

The metric is given here as the average on a per band basis to determine the integrity of the restored spectral bands, indicating if the information that was not kept can be regarded as noise. A lower BQ score indicates a more natural image. The BRISQUE model used is the default model provided by MATLAB 2020a. To the best of the authors' knowledge, the BRISQUE model has not been used previously on hyperspectral imagery.

6.3.4 Noise Characterization of Data Sets

The limited value of retaining noise during compression encourages the characterization of the data sets under study. Thus, a noise estimate in terms of SNR is given in Figure 6.7, following the approach presented in Section 6.2.1.

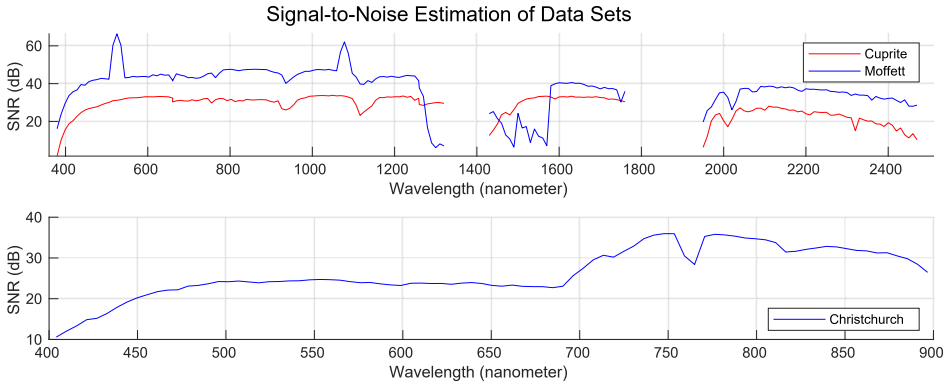


Figure 6.7.: Characterization of the SNR in the data sets, with SNR given as the log ratio of $\mathbf{X}_s$ variance over mean-square error when compared to $\mathbf{X}_o$, see section 6.2.1. The mean SNR is 27.67, 37.54 dB and 26.28 for Cuprite, Moffett, andHICOrespectively.

The data sets used have some intrinsic properties that characterize their appearance. As the AVIRIS dataset contains more elements, it is more complicated to estimate the targets' albedo. For the Christchurch data set from HICO, where the expected albedo would be about 5-10 percent, the reported SNR does not exceed 25 dB [192], and the mean SNR from Figure 6.7 is 26.28 dB. Still, according to [158], the dark current SNR does not exceed 35 dB. The mean SNR found here in Figure 6.7 is 27.67 dB for Cuprite and 37.54 dB for Moffett.

Following the justification presented in [166] the compression retains the information when the compression artifacts, here in terms of SNR, are less than the sensor noise. Beyond that, the hyperspectral data's end-user determines the acceptable compression artifacts. It should be noted that when the pre-computed loadings are tailored towards a specific application, e.g. modeling specific phenomena, a low SNR does not necessarily indicate a high loss of information.

6.4 Results and Discussion

6.4.1 Stage One

Stage One represents the baseline used for comparison with some variations of PCA in combination with JPEG2000. In this section, the baseline is also compared with just using JPEG2000. Results from the compression after stage one in terms of SNR, SAM, and BQ are given in Table 6.2. In this

table, the first column specifies the set bit rate, and performs in terms of SNR, SAM and BQ-score for the four modes when applied to Cuprite, Moffett and Christchurch, respectively.

The number of components from the loadings in the modes using PCA for dimensionality reduction corresponds to the best performance regarding the SNR metric for a given bit rate. The SNR metric was chosen as the optimization constraint as it has been stated to be a good indicator of classification performance [172], and is given the most attention in subsequent sections. Using the results from SAM as an optimization constraint gave virtually identical results for higher bit rates, but a difference could be observed at lower bit rates. These loadings are computed using testing sets described in Section 6.3.1 and illustrated in Figure 6.5 and 6.6.

The SNR and SAM performance is highest and stable for the PCA modes using increased components. However, using a higher number of components increases the processing time. This was observed for all scenes, and is further elaborated in subsequent sections.

In scenes with a high level of spatial details, such as the Moffett and Cuprite scenes, dimensionality reduction by PCA improves the compression for all tested metrics. This is in agreement with [165, 170, 171]. However, the studies mentioned above do not seem to include spatially homogeneous scenes, as the Christchurch scene given here is intended to represent. Similar scenes are representative of ocean color scenes, an important application of remote sensing [1, 196]. At higher bit rates, plain JPEG2000, i.e., J2K2D and J2K3D, does not achieve a compression performance close to PCA2D and PCA3D, which pre-processes the data with PCA before JPEG2000. However, for the Christchurch scene, J2K2D and J2K3D outperform PCA2D in SNR and SAM for high bit rates. This is not observed at lower bit rates. Thus, at higher bit rates, the cost of computing and projecting the loadings could, in time-constrained applications, become a factor for such scenes. This is not further explored within this body of work.

It is unclear whether a 3D or 2D encoding scheme for the JPEG2000 part is used in [170, 171] when combined with spectral decorrelation. Here, a 3D vs. 2D encoding scheme is compared in the JPEG2000 part of the first stage of the processing pipeline, both with and without spectral decorrelation. To exploit the three-dimensional structure yields slight performance benefits at higher bit rates when compressing HSIs with and without spectral decorrelation. This advantage diminishes as the compression is constrained to use fewer bits, and becomes destructive at lower bit rates, supporting findings from [179]. This is most likely due to incorrect assumptions regarding the 3D structure of HSIs [179, 197]. Furthermore, the BQ-score seems to favor both using a 2D-conversion for the JPEG200 encoding, and the effects from spectral decorrelation. This suggests that both 2D-conversion and dimensionality reduction by spectral decorrelation have denoising effects compared to the other compression modes in terms of the BQ-score.

For the higher bit rates, the data distortion from compression is similar to or lower than the estimated sensor noise from Section 6.3.4. As can be observed from Table 6.2, for the Moffett scene, the degradation did not exceed the effect of the estimated sensor noise in terms of SNR for 0.5 *bpppb*

or higher, 0.25 or higher for the Cuprite and Christchurch scene. To restore the data at this level further advocates for using few components and a lower bit rate when loss can be accepted.

Table 6.2.: Compression results, using online computed loadings. The parameters for the number of components in the dimensionality reduction and the JPEG2000 compression ratio have been optimized for the SNR metric.

bpppb	method	Cuprite			Moffett			Christchurch		
		SNR	SAM	BQ	SNR	SAM	BQ	SNR	SAM	BQ
4.0	J2K3D	40.23	0.14	44.3	36.63	0.56	44.08	51.98	0.06	35.69
	PCA3D	**53.94**	**0.03**	**44.28**	**62.49**	**0.03**	41.12	**54.65**	**0.05**	36.27
	J2K2D	38.77	0.14	44.38	36.59	0.56	46.1	51.47	0.07	**35.66**
	PCA2D	52.25	0.04	44.31	61.57	0.03	**41.1**	48.37	0.1	36.39
2.0	J2K3D	29.17	0.44	**44.09**	25.17	1.98	61.92	39.31	0.24	42.41
	PCA3D	**43.29**	**0.1**	44.27	**53.59**	**0.08**	41.26	43.73	0.17	33.9
	J2K2D	28.48	0.47	44.32	24.86	2.05	51.19	40.64	0.23	55.52
	PCA2D	42.6	0.11	44.3	52.53	0.09	**41.08**	**43.96**	**0.16**	**31.97**
1.0	J2K3D	23.75	0.69	43.19	19.05	3.72	57.87	33.59	0.44	38.6
	PCA3D	**38.35**	**0.18**	**44.32**	44.39	0.23	41.24	38.99	0.28	18.35
	J2K2D	23.12	0.72	43.05	18.7	3.9	45.63	35.6	0.39	52
	PCA2D	37.96	0.18	44.32	**47.38**	**0.16**	**40.97**	**39.39**	**0.27**	**15.54**
0.5	J2K3D	20.24	0.88	47.53	15.4	5.25	48.2	31.72	0.53	39.7
	PCA3D	**34.77**	**0.26**	**44.34**	38.35	0.46	41.68	36.19	0.39	26.46
	J2K2D	20	0.89	48.6	15.14	5.41	47.03	30.8	0.57	56.13
	PCA2D	34.48	0.27	44.39	**40.29**	**0.36**	**41.53**	**37.53**	**0.33**	**15.5**
0.25	J2K3D	17.23	1.02	60.54	13.06	6.39	47.5	27.08	0.73	50.36
	PCA3D	**32.03**	**0.36**	44.23	32.52	0.86	41.69	**36.7**	**0.37**	**15.71**
	J2K2D	17.78	1.07	45.44	12.94	6.44	46.43	26.05	0.79	49.23
	PCA2D	31.88	0.37	**44.09**	**33.21**	**0.79**	**41.64**	36.41	0.38	25.01

Table 6.3.: Compression results, using online computed loadings. The parameters for the number of components in the dimensionality reduction and the JPEG2000 compression ratio have been optimized for the SAM metric.

bpppb	method	Cuprite			Moffett			Christchurch		
		SNR	SAM	BQ	SNR	SAM	BQ	SNR	SAM	BQ
4.0	J2K3D	40.23	0.14	44.3	36.63	0.56	44.08	51.98	0.06	35.69
	PCA3D	**53.94**	**0.03**	**44.28**	**62.49**	**0.03**	41.12	**54.65**	**0.05**	36.27
	J2K2D	38.77	0.14	44.38	36.59	0.56	46.1	51.47	0.07	**35.66**
	PCA2D	52.25	0.04	44.31	61.57	0.03	**41.1**	48.37	0.1	36.39
1.0	J2K3D	23.75	0.69	43.19	19.05	3.72	57.87	33.59	0.44	38.6
	PCA3D	**38.35**	**0.18**	**44.32**	44.39	0.23	41.24	38.99	0.28	18.35
	J2K2D	23.12	0.72	43.05	18.7	3.9	45.63	35.6	0.39	52
	PCA2D	37.96	0.18	44.32	**47.38**	**0.16**	**40.97**	**39.39**	**0.27**	**15.54**
0.50	J2K3D	20.24	0.88	47.53	15.4	5.25	48.2	31.72	0.53	39.7
	PCA3D	**34.77**	**0.26**	**44.34**	38.35	0.46	41.68	36.19	0.39	26.46
	J2K2D	20	0.89	48.6	15.14	5.41	47.03	30.8	0.57	56.13
	PCA2D	34.48	0.27	44.39	**40.29**	**0.36**	**41.53**	**37.53**	**0.33**	**15.5**
0.10	J2K3D	13.18	1.19	65.01	11.04	7.23	45.92	22.03	1.14	46.25
	PCA3D	**29.1**	**0.5**	43.03	25.22	1.89	37.93	**35.9**	**0.4**	**17.28**
	J2K2D	15.12	1.23	45.91	10.98	7.32	45.79	20.5	1.24	47.26
	PCA2D	28.23	0.56	**41.24**	**26.33**	**1.64**	**37.28**	35.47	0.42	32.07
0.05	J2K3D	13.13	1.31	56.96	9.9	7.74	45.92	18.91	1.57	46.81
	PCA3D	26.82	0.63	32.85	20.37	3.14	37.88	23.78	1.64	35.19
	J2K2D	13.75	1.35	46.59	9.85	7.75	45.61	18.34	1.57	47.61
	PCA2D	**26.94**	**0.63**	**30.57**	**22.12**	**2.53**	**33.04**	**34.33**	**0.46**	**30.68**

6.4.2 Stage Two

From Table 6.2 the PCA2D mode, which benefits from both the dimensionality reduction by PCA and a purely spatial wavelet transform by JPEG2000, is the most desirable when varying both bit rates and scenes. This mode is therefore selected for further analysis in stage two.

Furthermore, the total bit rate for stage two is fixed to 4.0 *bpppb* for the results given in Figure 6.8 and 6.9. The top row for all stage two heatmaps represents the case when no residuals are included, i.e., only doing stage one, the baseline compression, with a 4.0 bit rate. This bit rate is intended to represent a high yet sensible lossy compression ratio. The *bit rate of residuals*-axis refers to the bit rate allocated to represent the residuals, e.g., when a bit rate of 3.6 is allocated for the residuals, a bit rate of 0.4 is used for stage one in Figure 6.4, always resulting in a total bit rate of 4.0 *bpppb*. Results from the compression after stage one and stage two in terms of SNR and computational time are given in Fig 6.8 as six heatmaps for Moffett. Figure 6.8 illustrates how the performance is affected by keeping all the residuals versus analyzing them before compression. The observed performance patterns are also found for SAM, and the discussed relationships are found across all data sets. For the *Relative Time* reported in Figure 6.9 the computational time from the instance from stage one with the highest SNR is divided by the computational time from Stage Two. For

completeness Fig 6.10 is given, representing the same as figure 6.9, but with a fixed bit rate of 1.0 *bpppb*.

For the results with residuals the loadings are computed using training sets as described in Section 6.3.1 and illustrated in Figure 6.5 and 6.6. The loadings are derived using a similar data set to what it is compressing. The loadings are computed using a training set to simulate the use of pre-computed loadings. In real-world applications, it would be sensible to perform some cross-validation of the pre-computed loadings to ensure that they are robust against the expected variations in the data that they are supposed to transform [143]. These pre-computed loadings are desirable to use to reduce the overall computational time [176].

In Figure 6.8, heatmaps related to stage one of the proposed pipeline are given for completeness and comparison. There is a clear trend of an increased computational time with an increased number of components used in stage one, as one could expect. This increased computational time is due to more pixels being encoded by the JPEG2000 stage [171], and an increase in either the spectral or spatial dimension would affect the computational time similarly. The computational time is not significantly affected by the target bit rate. That is, the performance in terms of SNR is enhanced by using more components, but it comes at a higher computational cost. Also, this enhanced SNR for higher bit rates is more significant at a higher number of components and negligible at a lower number of components. Thus, fewer components from spectral decorrelation support lower bit rates, providing a higher throughput at a reduced bit rate with very similar quality in terms of SNR and SAM for stage one of the compression. A higher bit rate should be allocated for the residuals from stage two, with lower bit rates for stage one.

In Figure 6.8 two scenarios are presented. That is, using all the residuals, and only significant residuals are determined by analyzing their distributions. The *ALL* label refers to the analysis using all the residuals. Residual analysis *1STDV* is defined as deeming the residuals that are outside one standard deviation of a Gaussian normal distribution as significant. The results given in the heatmaps advocate again for using lower bit rates when relying on fewer components for stage one, i.e., a lower bit rate should be assigned to be used for the compressed image cube bitstream. Albeit results for stage two using *ALL* residuals provide an improvement in terms of SNR, the magnitude of the improvement is not large enough to be shown in the heatmap with the given number of significant digits. Furthermore, encoding all the residuals is more computationally demanding than determining and encoding only the significant residuals. This result does not encourage the separation of systematic variation patterns and noise variations when all of the residuals are given the same priority. This is to some extent also demonstrated in [178], and as such, can be seen as an expected result.

Residual analysis *1STDV* increases the performance in terms of SNR when relying on pre-computed loadings, i.e., the SNR metric can be consistently increased at lower bit rates and fewer components.

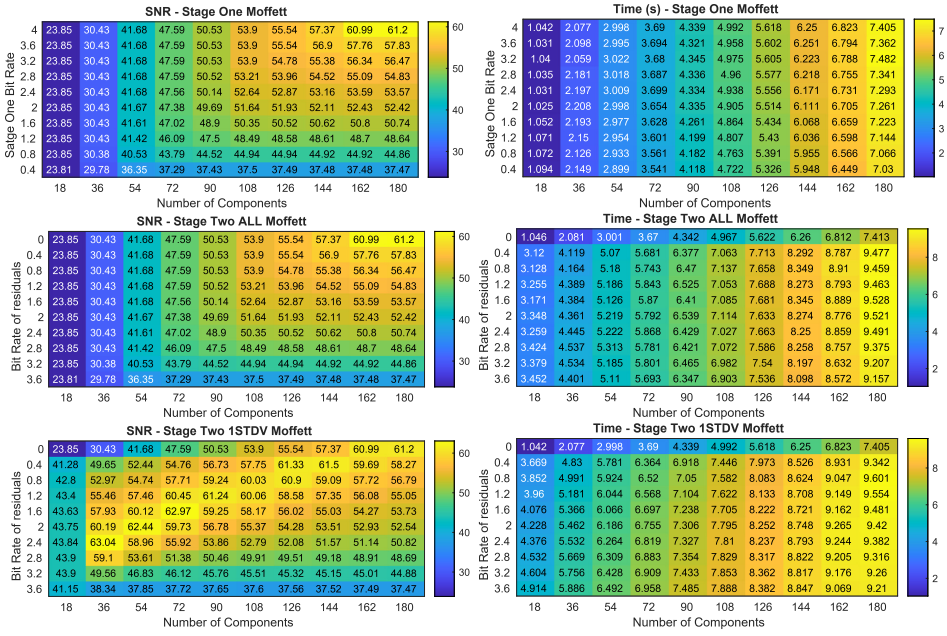


Figure 6.8.: Results for stage one and stage two in terms of SNR and Time for the Moffett scene. In stage two, the total bit rate is fixed at 4.0. This shows how the number of components used affects the SNR and computational time when different bit rates are used for the original image cube and residuals.

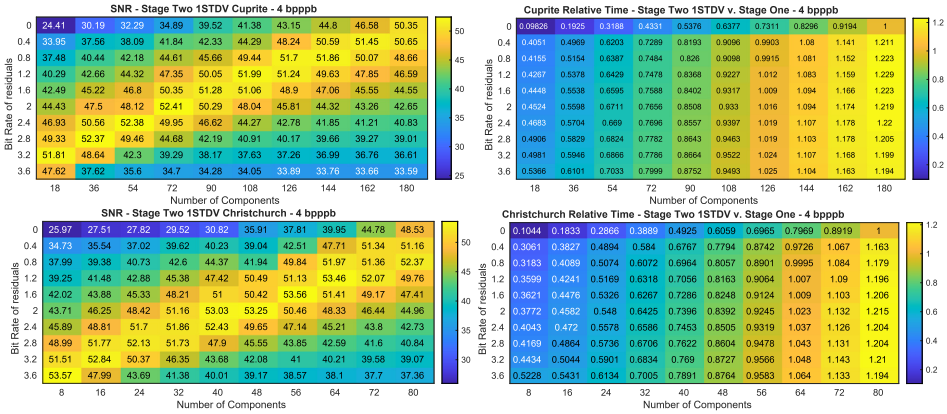


Figure 6.9.: Difference between stage one and stage two in terms of SNR and Time for the Cuprite andHICOscene. The bit rate is fixed at 4.0 *bpppb*. This shows how the number of components used affects the SNR and computational time when different bit rates are used for the original image cube and residuals. The top row where the bit-rate for residuals is zero equals only applying stage one.

For certain combinations of bit rates and the number of components, the combination of the original cube and the *significant* residuals can increase the performance in terms of SNR, when relying on pre-computed loadings. This was observed consistently across all data sets. These pre-computed loadings would perform for more drastically changing scenes or different viewing geometries is not investigated here. The performance is expected to degrade when relying on pre-computed loadings in these conditions. A more intriguing observation is that in some cases, the proposed method, in terms of SNR, outperforms the results reported in Table 6.2, where the loadings were computed online, even without including the loadings in the total *bpppb*. The online computed loadings are expected to perform better as they are (over)fitted to the specific data they are supposed to represent and compress. This is consistently observed across all data sets when relying on the PCA2D mode. Regardless, this supports the notion that depending on PCA-based spectral decorrelation has some limitations in retaining unexpected, unmodeled or poorly modeled phenomena, even when computing the loadings online using all the available data.

The SNR improvement that can be observed on the bottom-left to top-right diagonals of figure 6.9, is because most of the loss is coming from JPEG2000 when using more components and the HSI is ill-represented using high bit rates and few components. As the residuals come from the dimensionality reduction, they cannot account for the loss that occurs due to the wavelet transform in the JPEG2000 encoding scheme. An attempt at encoding and decoding the JPEG2000 transformed image was performed, but this put a too big computational strain on the processing pipeline to be justified. Accounting for the loss in the spatial wavelet-transform better is a topic for future research.

The performance patterns found with a fixed bitrate of 4.0 *bpppb* presented in Figure 6.9 does not appear in Figure 6.10 where a fixed bitrate of 1.0 *bpppb* is used. That is, for a fixed bit rate of 1.0 *bpppb* the proposed processing pipeline only functions as a reassurance measure to know that no significant unmodeled phenomena were lost due to using lossy compression. However, if retaining the data in terms of SNR is the main objective of the lossy compression, stage two should not be included in this scenario. With a total bit rate of 1.0 *bpppb*, no clear performance patterns in terms of SNR across all scenes are observed. Nonetheless, for the spatially homogeneous scene of Christchurch, where the wavelet-based JPEG2000 is expected to distort less at lower bit rates, it appears beneficial to include stage two of the proposed pipeline. Regrettably, for a bit rate of 1.0 *bpppb*, the distortion from the lossy quantization performed by JPEG2000 after the residuals does not well compensate for the wavelet-transform for the spatially complex scenes, here represented by the Moffett scene. This is expected as the residuals only compensate for the projection done by the loadings. A metric of the spatial complexity could be used as a threshold for determining whether stage two should be utilized, and the BQ-score is a candidate for such a measure. Again, this indicates that it would be beneficial to compensate for the distorting effects from the lossy wavelet transform for the best possible result. Some of the limitations briefly mentioned in Section 8.2 could help elevate the performance at this lower bit rate as well.

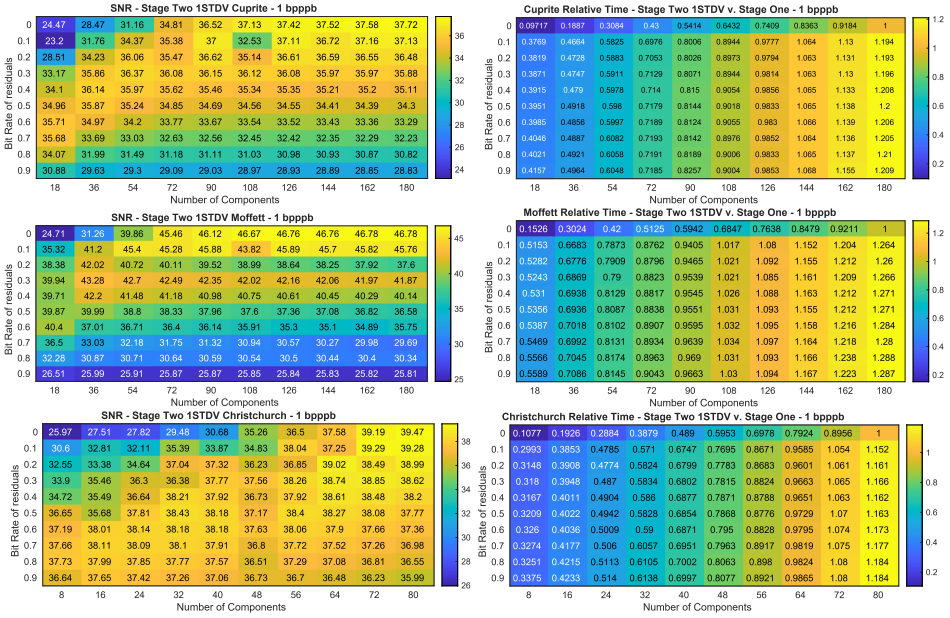


Figure 6.10.: Difference between stage one and stage two in terms of SNR and Time for all scenes. The bit rate is fixed at 1.0 *bpppb*. This shows how the number of components used affects the SNR and computational time when different bit rates are used for the original image cube and residuals. The top row where the bit-rate for residuals are zero is equal to only applying stage one.

A criterion or procedure to derive the optimal combination for stage two of the number of components used and bit rate for residuals needs further investigation. An approximate procedure, referred to as the *ratio criterion*, is given here. An observed trend from Figure 6.8 and 6.9 for an approximate procedure is given as the percentage of components that are used should be the inverse of the ratio of the bit rate allocated for the analyzed residuals. When 10% of the components are used 90% of the bit rate should be used for the residuals, and if 90% of the components are used 10% of the bit rate should be used for the residuals. Results from using this criterion to select the ratio are given in Figure 6.11. In this figure, online PCA is used for all the denoted components, with the computational time and SNR for this approach. From this figure, it is clear that the *ratio criterion* does not consistently provide the best relationship between the number of components used and bit rate for residuals. Nevertheless, it does show that the computational time can be reduced without much distortion in terms of SNR when relying on fewer components in stage one. Thus making it possible to rely on few components for lower bit rates in stage one as an initial step for data transmission or storage. In Figure 6.11 it is demonstrated once again that the proposed approach is best suited for spatially homogeneous scenes such as the Christchurch scene.

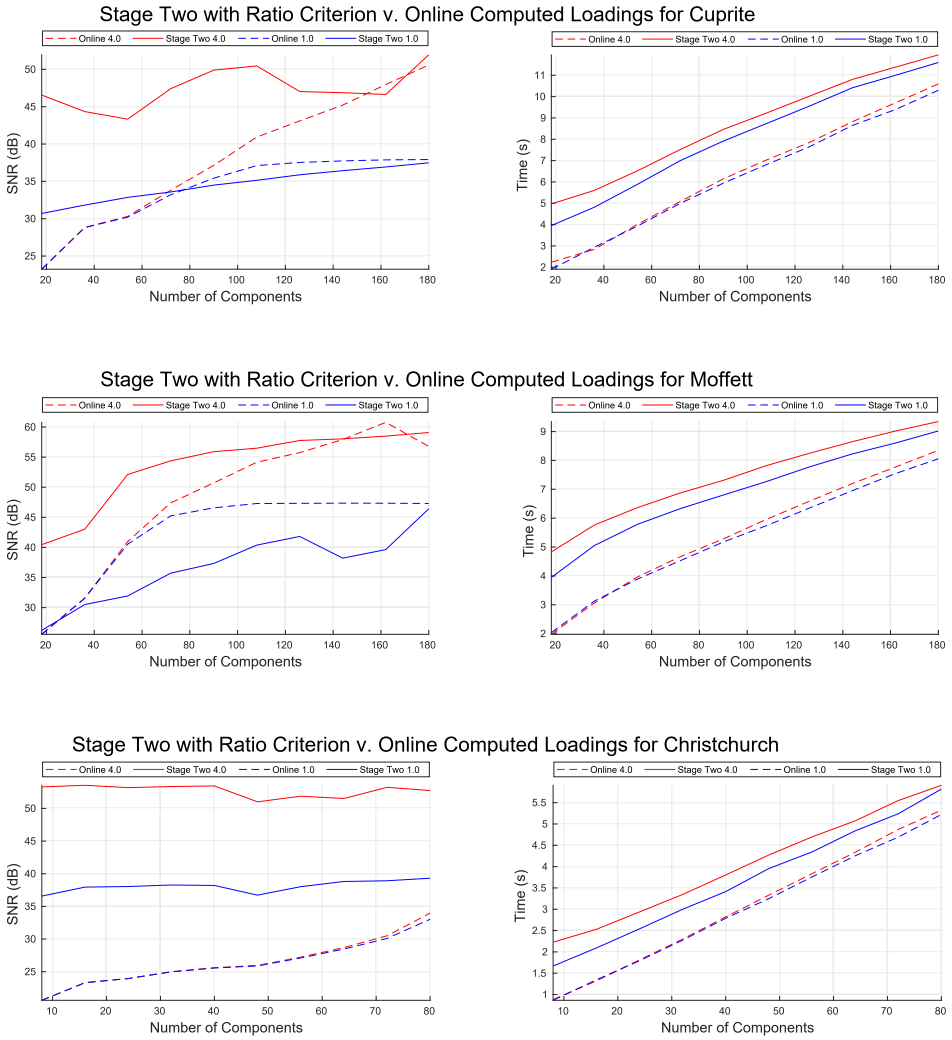


Figure 6.11.: Here PCA2D with online computed loadings for stage one without residuals is compared with The computational time and SNR of stage two PCA2D with pre-computed loadings. The bit rate indicated in in the legends refers to the total bit rate.

6.4.3 Performance of the Compression and Comparison

The resulting distortions from compression are non-destructive if the compression artifacts in terms of SNR (or SAM) are less than the expected sensor noise. That is, an argument can be made stating that the compression can be considered near-lossless if its artifacts are less than the effects of the expected sensor noise found in the original data set [166].

In [166, 170], the compression is demonstrated to be near-lossless by comparing target detection performance before and after compression. This can be misleading as transformation-based compression often attempts to reduce noise and can improve the target detection performance [80, 168, 172]. Reducing the noise, i.e., the uninformative variability in the data, can enhance the exploratory algorithms, and an improvement in target detection performance does not guarantee that the data fidelity is retained for other types of HSI applications, such as anomaly detection [172]. Dimensionality reduction can reduce the noise in the original data set by means of prioritizing data that contain systematic variation patterns, but will have more significant errors on anomalous data that will appear as larger residuals. It is also observed that the BQ-score favors the use of a low number of components in dimensionality reduction and low bit rates. The results presented in Table 6.2 show that the distortion in terms of SNR and SAM for the first stage. These results are in alignment with results reported in [166, 170, 171, 179] for similar methods. The stage one compression performance in terms of SNR and SAM, combined with the BQ-score, clearly indicates denoising effects by reducing the number of components. Combining these metrics, with some constraints of acceptable ranges for SNR and BQ-score, can be a practical regularizing objective for *optimizing* lossy compression.

Limitations of online usage of PCA can are mitigated by estimating the loadings from similar data as performed here. Furthermore, it is possible to compute the loadings by sub-sampling the data set as in [80, 165, 172, 176]. Also, nonlinear pattern variations, i.e., variations that cannot be represented well by a linear transform, in the data can be compensated for by pre-processing using kernels before PCA [4, 180], but this comes at an additional computational cost. PCA can preserve general trends in the data in terms of SNR, but can fail to account for anomalies [172]. These anomalies can be better preserved by including residuals, preferably significant ones. This approach provides an overview of the nature of the residual data excluded from the compression. This overview about what is lost during the compression allows low bit rates in stage one, the lossy compression, with added confidence in the data fidelity. With the insight from residual analysis, the significant residuals can be retrieved without loss [165] or in a lossy manner, as proposed here, to retain control of the total bit rate.

As the proposed approach only compensates in stage 2 for the distortions from PCA projection, scenes that are more effectively encoded by the latter JPEG2000 stage will benefit more from using it. This is shown here in multiple ways for the Christchurch scene.

6.4.4 Analysis of Residuals

From Figure 6.8, it is clear that if residuals are used to compensate for model inaccuracies, it is beneficial to perform the analysis before the compression of residuals. There is an added computational time for analyzing the residuals before compressing them. However, the results from stage one can then be compressed using less time due to fewer components being used, and fewer pixels are encoded at a similar level of distortion. As illustrated in Figure 6.9, for certain combinations of few components and low bit rates for stage one with a higher bit rate for analyzed residuals, a higher SNR can be achieved at a reduced relative computational time that our implementation produced. Furthermore, the analysis of residuals is decoupled from compressing HSI after PCA-based dimensionality reduction. Thus, after projection, the residuals can be processed separately from stage one. Here, the residuals that exceed one standard deviation of an estimated Gaussian normal distribution are considered significant. However, other approaches to determine significant residuals can be considered dependent on constraints such as computational time and data fidelity. Examples of alternative approaches are to only transmit residuals that exceed a given threshold, or only transmit a fraction of the most significant residuals in terms of a given norm. The proposed *1STDV* approach used here gave the most desirable results in terms of computational time and data fidelity in general.

A simple addition to the residual analysis presented is to generate aggregated statistics, i.e., information about the distribution of residuals. This could, and perhaps should, be transmitted before the residual bitstream. This meta-information can then aid in determining if the residuals are significant and if they should be transmitted at all.

6.4.5 Datatype Conversion

There are some significant limitations in our implementation. The input HSI data used here is stored as 16-bit unsigned integers. The dimensionality reduction and wavelet transformations are performed using 64-bit doubles. The original data was extrapolated from its original bit representation size to doubles.

When performing the processing steps, which use floating-point operations, the higher bit resolution mitigates the loss of precision. This is at the cost of using more of the available computational resources. The last processing steps of quantization and source coding are needed to reduce the data volume below the original data size after this data type conversion. As discussed, the quantization strategy used could and should be improved upon, but this is not straightforward in the given implementation. This discussion point confirms the analysis in [165, 171, 173], but in this chapter, the effect of a different bit representation for other components have not been evaluated. Future research is subject to future research if the processing pipeline is implemented on resource-limited hardware. Additionally. the proposed compression approach can be made sequentially, on parts of a cube, to address limitations regarding limited computational resources.

6.4.6 On the Use of 2D Wavelets

Wavelet transform performance depends on the given characteristics of a scene, i.e., the smoothness, sharpness, and noise of an image cube affect how the wavelet transform contributes to the compression performance [197]. As HSIs are inherently three-dimensional, the characteristics may differ within a purely spatial or spatial-spectral frame of the cube and in the remaining dimension, and this is not accounted for in the 3D-based compression provided by JPEG2000 Part-10 [179].

A 3D domain transform that leverages the entire structure of the image cube may not provide the best performance as the correlation of spatial or spectral pixels within a particular dimension is different from that between slices [197]. The resolution, either spatial or spectral, dictates the expected correlation. Thus, choosing a wavelet filter optimal for the spectral dimension may not be optimal in the spatial dimension, and vice versa [170, 179, 197].

Even though JPEG2000 in its Part-10 is designed to provide true 3D coding, it is still oriented toward data that is isotropic in all directions, i.e., has a spatial interpretation in all directions [179, 197]. This assumption does not fit well for hyperspectral remote sensing imagery where one dimension is spectral. An HSI will not be as effectively coded with the same underlying assumptions. The use of a 3D wavelet(s), potentially altering wavelets to accommodate for the spectral dimension, can improve the performance and is a topic for future research. The multi-component transform can also be used within the JPEG2000 framework to alleviate this.

If using video format encoding with dimensionality reduction by PCA, it is also essential to embed that scores from low eigenvalues are less significant than the scores from high eigenvalues. Furthermore, some blocking strategies used in video encoding will not retain the data fidelity in HSIs, as it will prioritize human appeal over image restoration fidelity. To some extent, this is handled by performing 2D-conversion at the cost of not exploiting repeated patterns in the *spectral* dimensions of the score cube.

To better resemble regular spatial images, 2D-conversion is used to be able to utilize the well-developed 2D image compression of JPEG2000 fully for natural images [180, 189, 190].

6.4.7 Computational Time and Considerations for Real-time compression

The computational time presented in Figure 6.8 and 6.9 has some caveats. The presented results originate from a MATLAB 2020b implementation on a desktop computer with an i7-8650U CPU at 1.90GHz running an Ubuntu 20.04.2 LTS operating system. The results shown here are not optimized for computational time concerns, and at their current state, are not suited for real-time compression on any platform. This is a topic for future research. It is not straightforward to compare with other proposed processing pipelines found in the literature [170, 171, 176], but the baseline ran on the same system is intended to be a proxy for comparison. However, the processing protocols from stage one are shown in [176] to perform real-time compression on a Virtex-7 FPGA when

considering a sensor output of 491.52 Mb/s. This is even achieved when computing the loadings online, albeit using a sub-sampling strategy. In addition, the implementation given in [171, 176], also uses vector quantization to further compress the scores, at an additional computational cost.

The number of floating-point operations needed to conduct PCA is expected to increase with larger HSI cube sizes. According to [177] the number of floating-point operations will increase with the power of three for the spectral dimension and with the power of two for the spatial dimension. If pre-computed loadings are used, there is mainly a projection stage which will increase linearly [177]. This projection stage can be done independently of other parts of the data.

As the residual analysis is decoupled from stage one after projection, and consists of virtually identical processing steps, it should be possible to achieve a similar performance given that sufficient programmable logic is available. After the projection, the residual analysis can be done in parallel. However, further research is needed to make the proposed pipeline operationally viable.

There are several techniques to reduce the run-time requirements, including the use of Window PCA, OTFP, or similar techniques [84, 172, 174]. If the models are built offline, it is only required to transmit the compressed results. By analyzing the residuals, and transmitting these by priority, one can use parts of the available bandwidth to learn from the data that is not well represented by the computed loadings. This ensures both a high level of compression, and the opportunity to improve the transformation models over time [84, 184].

6.5 Conclusions

Different approaches for transform-based encoding and compression using bi-linear modeling by PCA of known spectral variation patterns and wavelets greatly improved the overall compression performance when compared to just using JPEG2000 alone, as expected [170, 172]. This property became even more prevalent at lower bit rates. The metrics used to measure data integrity, from the original to the restored, indicate that its data volume can be significantly reduced without significant loss of information. These metrics suggest that the transform-based encoding has some denoising effects, allowing compressing the HSIs using a lower bit rate.

For HSI remote sensing applications where it is not desirable to use resources to recreate and analyze the noise in the acquired data sets, a near-lossless compression scheme should be considered. By decoupling the informative data and the PCA model discrepancies, it is possible to transmit the most significant trends in the original data at relatively low bit rates while retaining the unmodeled or poorly modeled phenomena as residuals. Further, by analyzing the residuals, and only transmitting those deemed significant, the performance can be improved in terms of both restoration ability and computational time. The residual analysis provides greater confidence in lossy compressed data, and illuminates where the transformation-based encoding can be limited. The proposed pipeline increased SNR of up to 2 dB at 77% of the computational time for the Cuprite scene. An increase in SNR was observed across all scenes, and the right combination of components and bit rates yielded a reduced overall computational time and higher SNR. The *ratio criterion* is suggested as

an aid in determining the bit rate allocated for the compressed image and residual bitstream, but does not provide an optimal solution. The improvement from utilizing stage two is made possible by needing fewer components to represent the data at the same or better quality than in stage one. Residual analysis can significantly reduce the number of elements that need to be encoded in the second stage. The proposed approach only compensates for distortions from PCA projection. Thus scenes that are more effectively encoded by the latter JPEG2000 stage will benefit more from using it.

With the addition of residuals filtered by residual analysis, higher performance in terms of SNR, SAM, and computational time can be achieved at high bit rates, here exemplified with 4.0 $bpppb$, for near-lossless hyperspectral image compression.

6.5.1 Future Work

No optimization scheme for the quantization is applied in work presented here, a potential improvement of compression ratio can be achieved by proposing an optimization scheme for the quantization at the cost of higher complexity.

The fidelity of the transform-based compression can be difficult to determine, different methods for regularizing the number of components, e.g., by BQ-score, and their limitations are topics for future research.

The use of nonlinear autoencoders, for decorrelation, as presented in [166], could outperform PCA-based spectral decorrelation and dimensionality reduction [170, 171]. A comparison accounting for the information and fidelity, as well as computational cost [166, 176] for decorrelating HSIs, offline or online, is a topic for future research. Further experimental investigations are needed to determine if other spectral decorrelation approaches, e.g., autoencoders, are better suited for decorrelating the spectral dimension of HSIs than PCA, and how the second stage is affected by the method for dimensionality reduction.

As JPEG2000 is an aging approach to image compression, some encoding advancement has come since then. As demonstrated in this chapter, the 2D conversion enables the use of all image encoders designed for natural 2D images, without putting many assumptions on the third dimension, both with and without spectral decorrelation. Even with the spectral correlation, while the compression is improved, it would be fair to assume that more modern encoding schemes can further compress natural images. Some modern encoders are designed for image fidelity rather than naturalness. This could fit the needs of hyperspectral remote sensing better. An investigation could result in a higher lossy compression ratio or lower bit rate for similar hyperspectral images.

Chapter 7

Atmospheric Correction Over Coastal Waters

We think we receive all that we perceive,
but in fact, we actually give the sky its colour.

James Turrell

Common Atmospheric Correction (AC) algorithms for ocean color attempt to predict water-leaving radiance and works well for the open-ocean using multispectral data. However, it can be inaccurate or computationally demanding for coastal and optically-complex waters, i.e., case II type waters [7, 60]. In this type of waters, the phytoplankton signal might be masked or modified by the presence of other substances, especially CDOM, and Total Suspended Matter (TSM). Coastal waters are of interest to the aquaculture industry, marine science, recreation, and environmental monitoring. Most of the measured top of atmosphere (ToA) radiance over coastal waters is comes from atmospheric contributions. Thus, retrieving valuable properties from the water-leaving radiance can only be done well if the atmospheric correction algorithms are accurate. From water-leaving radiance, it is possible to derive ocean color data products, such as maps of [Chl a]. These maps are, in turn, used to study phytoplankton changes in the ocean ecosystems to understand global climate change. Here, different Machine Learning models are presented, trained, and evaluated using **simulated** hyperspectral ocean color data of ToA radiance from coastal waters to predict water-leaving radiance and other ocean color variables directly, such as chlorophyll concentration. When trained and evaluated on **simulated** data, the proposed methods are shown to achieve an accuracy of up to 99%.

Atmospheric Correction (AC) has traditionally been treated as a preliminary step used to derive parameters of interest [1, 136]. However, some ocean-atmosphere interactions are not necessarily well understood, such as the effects of sea-spray on aerosols [198].

An understanding of the biogeochemistry, ecology, and hazards of the oceans with a changing climate is critical to sustaining Earth as a habitable planet [2]. Satellite remote sensing of the ocean's spectral albedo is an effective tool to characterize and monitor the ocean environment on a large or even global scale. Ocean Color is used to monitor, among other things, [Chl a]. This [Chl a] parameter is a common biomarker of photosynthetic phytoplankton which is used to infer about the state of the marine ecosystems and provides the aquaculture industry and government agencies with information regarding water quality, biogeochemical cycles, and as an aid in fisheries management. These decision-making processes are better aided by knowledge of the biotic signatures of the different ecosystems and the separation of those signals created by the atmosphere. Here, Radiative Transfer (RT) models will be utilized to characterize and separate the atmospheric signals and explore Machine Learning (ML) solutions to identify coastal ecosystems from their spectral albedo using simulated data. The goal of these models is to discriminate against the atmospheric transmission scenarios typically present for HSI data [2, 4].
Originally, the analysis was performed using ML models for Neural Network (NN), Stochastic Gradient Descent (SGD)-, PLS-, and Support Vector (SVR)-Regression. All models were used to train AC models on HSI data, but only the results from NN and PLS regression are presented in this chapter, as they were the most promising. These two cases represent both a non-linear and linear approach. The complete analysis with all ML models and a more in-depth discussion of the results can be found in [199]. To generalize well with ML one needs to obtain data representing an abundant number of various environmental scenarios. The approach presented here uses synthetic data sets due to the difficulty of finding large amounts of HSI data with corresponding metadata, such as sun-target-sensor angles alongside $in-situ$ measurements of light fields, spectral properties and pigment information for validation. Similar approaches using simulated data of multispectral radiance have been used to verify AC algorithms in [2,200,201]. The results from these publications will form a basis for comparison.

In this chapter, the RT model AccuRT [202] is used to simulate different variations of the expected HSI data in terms of radiance. Scenarios representative of a wide range of atmospheric and coastal oceanic environments, both strong aerosol containment and case II waters were simulated. The ML models were trained on the simulated data, to predict remote sensing reflectance ($R_{rs}(\lambda)$) from different ToA radiances. Also, retrieval algorithms for IOPs of water based on ML were produced, aimed to predict the commonly retrived IOPs from water [2, 60]. The ML models will predict [Chl a], mineral concentration (TSM) and the absorption coefficient at $443\,\mathrm{nm}$ for CDOM ($a_{cdom}(443)$) from $R_{rs}(\lambda)$, defined in Eq. (7.1). The trained ML models is then validated against each other concerning the accuracy, computational complexity, and interpretation capability, to study which could be suitable for on-board processing. Giving an indication as to how well the approach found in [200] for multispectral data works for HSI data, specifically for optically complex waters.

7.1 Problem Formulation

As most of the satellite measured ToA radiance over waters is due to atmospheric contributions, retrieving useful properties from the water-leaving radiance could only be done well if the AC algorithms are accurate. Only a relatively small portion of the incoming sunlight is backscattered from below the ocean surface in comparison with the sunlight backscattered from the atmosphere and specular reflection from the surface [9, 203].

With $L_w(0^+, \lambda)$ as water leaving radiance just above the sea surface and F_0 as the extraterrestrial solar irradiance, the $R_{rs}(\lambda)$ can be expressed as

$$R_{rs}(\lambda) \equiv L_w(0^+, \lambda) \,/\, \mathrm{F}_0 \cos(\theta_0)\, t_0(\lambda, \theta_0), \tag{7.1}$$

with the total measured ToA radiance at a given wavelength λ, $L_t(\lambda)$, for ocean-atmosphere systems can be expressed as the partitioned linear equation [111], see Figure 7.1.

$$\begin{aligned} L_t(\lambda) = L_r(\lambda) + L_a(\lambda) + t(\lambda)L_{wc}(\lambda) + \\ t(\lambda)L_{sky}(\lambda) + T(\lambda)L_{sun}(\lambda) + t(\lambda)L_w(\lambda), \end{aligned} \tag{7.2}$$

where $L_r(\lambda)$ is the radiance due to Rayleigh scattering by air molecules, $L_a(\lambda)$ is the aerosol scattering, $L_{wc}(\lambda)$ is the radiance contribution from whitecap on the sea surface, $L_{sun}(\lambda)$ is the specular reflection of direct sunlight off the sea surface, $L_{sky}(\lambda)$ is the radiance contribution from surface-reflected background atmospheric radiance and $L_w(\lambda)$ is water-leaving radiance due to photons that penetrate the sea surface and are backscattered. Diffuse and direct transmittances are given as $t(\lambda)$ and $T(\lambda)$, respectively.

Other AC algorithms, e.g FLAASH, ATREM and POLDER [4, 9, 200], are based on a computationally demanding RT model, like MODTRAN, or a set of pre-calculated spectral Rayleigh scattering values, stored in Look Up Tables (LUT), to compute $L_r(\lambda)$. According to [200], RT models can give an uncertainty lower than $0.5\,\%$ when predicting $L_r(\lambda)$, which is also the major contribution to $L_t(\lambda)$. The algorithms could retrieve values from the LUTs matching the geometry and parameters from a scene and use interpolation for values in between. Many AC algorithms for ocean color are based on the assumption that electromagnetic radiation in the NIR region back-propagated out of the water can be assumed to be zero, i.e., the black ocean assumption. This assumption is then used to estimate aerosol contributions.

While this approach works well for open oceans, it does not fit coastal areas where the black ocean assumption tends to fail, and the aerosols can be more optically complex. One method to address this problem was studied by [200], where the combined aerosol and Rayleigh-corrected TOA radiances for multispectral data were used together as input to a NN, where $R_{rs}(\lambda)$ was predicted. A similar approach is presented here for HSI data.

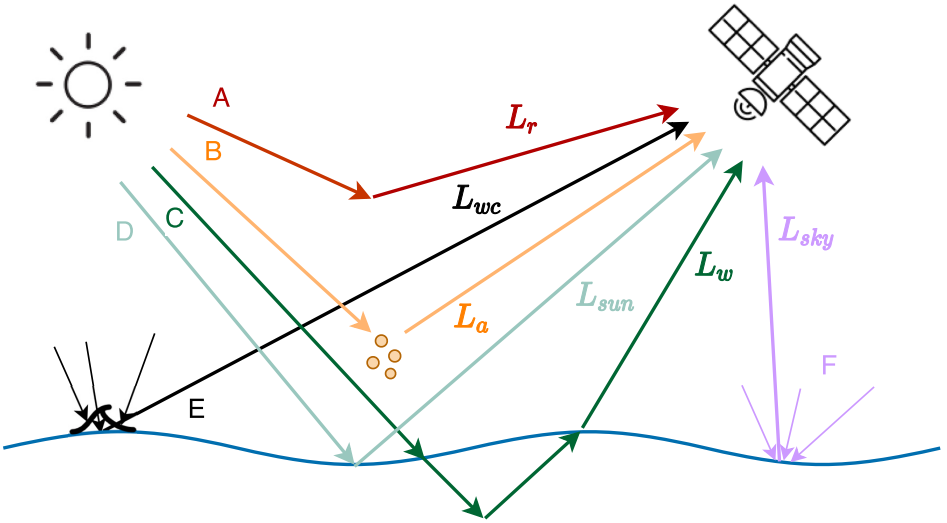


Figure 7.1.: Illustration of different contributions to the sensor-measured radiance. A, B, C, D, E, F refer to Rayleigh (L_r), aerosol (L_a), water-leaving (L_w), direct sun glint (L_{sun}), whitecap (L_{wc}) and sky glint (L_{sky}) radiance, respectively [9]. Illustration by Ole Martin Borge.

7.2 AccuRT Model

The coupled atmosphere-ocean RT Model AccuRT was used to simulate the interaction of solar radiation with particles and molecules in the atmosphere and ocean. AccuRT is a well-tested, user-friendly, and accurate radiative transfer model also capable of including effects from case II waters, and was used to simulate different cases of spectral radiance data representative for strong aerosol containment and case II waters [202]. AccuRT was used to generate synthetic data consisting of HSI ToA radiance and corresponding $R_{rs}(\lambda)$ for a large variation of aerosol and ocean body properties for 400-$800\,\mathrm{nm}$ wavelengths with $5\,\mathrm{nm}$ spectral sampling. This is the spectral range that HYPSO-1 is expected to operate in [5, 23]

7.2.1 Atmosphere and Aerosol

AccuRT uses a stratified vertical structure defined by the intensive properties of an atmosphere in hydrostatic balance. A 14-layer atmosphere covering the first $100\,\mathrm{km}$ was used with the predefined U.S. Standard atmospheric profile [200, 202]. The aerosols were added to the boundary level with Aerosol Optical Depths (AOD) at $869\,\mathrm{nm}$ chosen between 0.0001 and 0.4 was used. In AccuRT, it was not possible to specify AOD at any given wavelength directly, but the variation in AOD(869) could be included by varying the values of volume fraction of aerosols (f_v), the fraction of fine and coarse aerosols (f_s), and relative humidity (RH). Value ranges of these aerosol-specific parameters could be chosen to get AOD(869) values between 0.0001 and 0.4, shown in Figure 7.2. With θ_0,

θ, $\Delta\phi$ as solar and sensor zenith, and relative sensor azimuth angle, respectively, the ranges of simulated values are shown in Table 7.1.

Table 7.1.: Different input parameters used for the AccuRT simulations, their ranges and how the parameters were selected.

Parameter	Name	Value range	Unit	Selection
θ_0	Solar Zenith	0-65	$[^\circ]$	Uniform
θ	Sensor Zenith	0-70	$[^\circ]$	Uniform
$\Delta\phi$	Solar Azimuth	0-180	$[^\circ]$	Uniform
RH	Relative Humidity	30-95	[%]	Uniform
f_s	Fraction of Aerosols	0-1	unitless	Uniform
f_v	Aerosol Coarseness	1e-12 - 1e-10	unitless	Uniform
[Chl a]	Chl-a Concentation	0.006 - 98	$[\mathrm{mg/m^3}]$	Distribution
TSM	Temp	0.002 - 99	$[\mathrm{g/m^3}]$	Distribution
$a_{cdom}(443)$	Temp	0.0004 - 5	$[\mathrm{m^{-1}}]$	Distribution

7.2.2 Water IOPs

To simulate a representative synthetic dataset for the ML algorithms, water IOPs were extracted from $in\text{-}situ$ field measurements. The water IOPs extracted, and needed for AccuRT data generation, were concentration of $a_{cdom}(443)$, [Chl a] and TSM. Data were extracted from the NASA bio-Optical Marine Algorithm Dataset (NOMAD) dataset and Coast Color Round Robin (CCRR) datasets [204]. The distribution of [Chl a], TSM, and $a_{cdom}(443)$ are shown in Figure 7.2. The goal was to extract water IOPs representative for Case 1 and case II type waters. When generating data with AccuRT, the bio-optical model CCRR was used [202].

7.2.3 Data Generation with AccuRT

The main challenge for AC over coastal waters is that both the water and the aerosols are more optically complex than for open ocean conditions. This study will address this problem as done in [200] by looking at the simulated Rayleigh-corrected TOA radiance ($L_{rac}(\lambda)$), defined as:

$$L_{rac}(\lambda) = L_a(\lambda) + t(\lambda)L_w(\lambda), \tag{7.3}$$

where $L_{rac}(\lambda)$ is therefore the ToA radiance corrected for atmospheric gas absorption, Rayleigh, glint and whitecaps. These effects were removed as they often are corrected for in satellite image processing [200]. This means that some pre-processing would be needed before applying these models to an operational on-board processing scenario. The trained AC models would then predict $R_{rs}(\lambda)$ from $L_{rac}(\lambda)$,

For training and tuning/regularization of the ML algorithms, 85% of the total simulated data was randomly selected, using the regularization approach described in Section 7.3.2 and 7.3.3. The remaining 15% was used to test the models, with the results shown in Section 7.4.

7.3 Data Preparation & Machine Learning

7.3.1 Data Pre-processing

The spectral radiance input was divided by the cosine of the solar zenith angle, which is a term that can be found in Eq. (7.1), to get reflectance.

The Savitzky-Golay filter (Savgol) can be applied to a set of discrete data points to smooth the spectral data without distorting the signal tendency [205]. Different types of derivatives can be applied to the spectral Savgol filtered data. This filter has long been used for spectroscopy to smooth and differentiate absorption spectra [206]. As the filter improved performance, it was used as another pre-processing step.

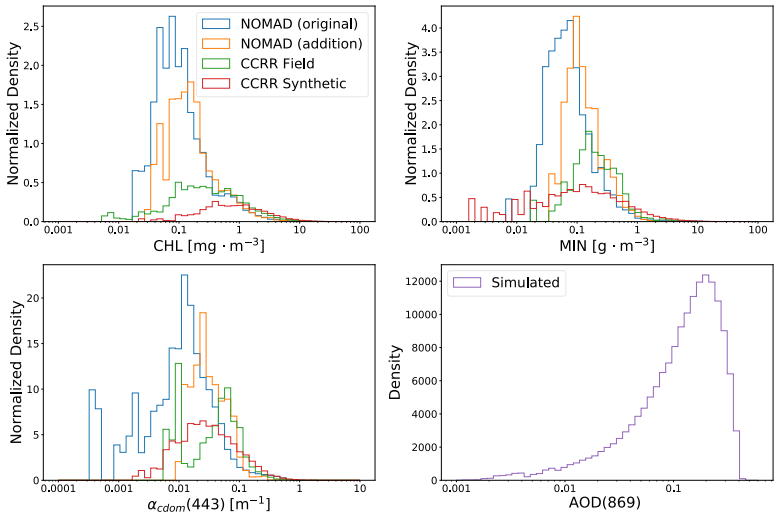


Figure 7.2.: Distribution of [Chl a], mineral concentration (TSM) and $a_{cdom}(443)$ extracted from different field datasets, together with the distribution of AOD(869) generated with AccuRT. Illustration by Ole Martin Borge.

Finally, the input data was normalized as given in Eq. (7.4), where $\hat{\mathrm{X}}$ is the normalized data, X is the original input data, $\overline{\mathrm{X}}$ is the mean and σ_X is the standard deviation.

$$\hat{\mathrm{X}} = \frac{\mathrm{X} - \overline{X}}{\sigma_X} \tag{7.4}$$

7.3.2 Sequential Neural Networks for Regression

It has been demonstrated that NNs with one or more hidden layers can predict non-linear relationships that could be suitable for deriving remote sensing reflectance $R_{rs}(\lambda)$ from various ToA radiances [9, 200]. The NN presented here is a simple feed-forward NN, also known as the multilayer perceptron. The Python Deep Learning library Keras, with TensorFlow as a backend, was used to build the NN models, which were simple sequential models [207]. Several variations were tested, and the NN presented here used the "adam" optimizer with 2 hidden layers, 700 neurons in the first hidden layer, ReLU as activation function, and MSE as cross-validation metric for tuning/regularization during training. Again, more details can be found in [199].

7.3.3 Partial Least-Squares Regression

PLSR reduces collinearity and noise within a given dataset by iteratively relating data matrices using linear multivariate models. It is a two-step algorithm that first finds uncorrelated components in the variables of a given data set and then performs the least squares regression on these components. A more in-depth description of the algorithm can be found in [208]. Several variations were tested, with 10-fold-cross-validation for tuning/regularization, and with variable selection [207].

For IOP prediction, 81 bands and 22 components gave the best results, as shown in Figure 7.3. However, varying these hyperparameters did not yield very different results [199].

7.4 Results

The results of AC and IOP prediction using the different ML models were compared using the Pearson correlation coefficient (R), the average percentage difference (APD), and the normalized root mean squared difference (NRMSD), described in Eq. (7.5), (7.6) and (7.7), respectively.

$$\mathrm{R} = \frac{1}{N}\sum_{i=1}^{N}\left(\frac{Y_i - \bar{Y}}{\sigma_Y}\right)\left(\frac{\hat{Y}_i - \hat{\bar{Y}}}{\sigma_{\hat{Y}}}\right) \tag{7.5}$$

$$\mathrm{APD}\ [\%] = \frac{1}{N}\sum_{i=1}^{N}\left|\frac{Y_i - \hat{Y}_i}{\hat{Y}_i}\right| \times 100 \tag{7.6}$$

$$\mathrm{NRMSD} = \frac{\sqrt{\frac{\sum_{i=1}^{N}(Y_i - \hat{Y}_i)^2}{N}}}{\hat{Y}_{max} - \hat{Y}_{min}} \tag{7.7}$$

Here N is the number of samples, Y_i is the i-th predicted radiance value at a given wavelength, $\hat{Y}_i$ is the corresponding ground truth radiance value, σ_Y and $\sigma_{\hat{Y}}$ are the standard deviation of Y and $\hat{Y}$, $\bar{Y}$ and $\hat{\bar{Y}}$ are the mean values of Y and $\hat{Y}$, and $\hat{Y}_{max}$ and $\hat{Y}_{min}$ are the maximum and minimum value of $\hat{Y}$, respectively.

The ML models would give 81 predicted outputs from the wavelength bands between 400 and $800\,\mathrm{nm}$. Metric values for each predicted wavelength band were calculated and would therefore yield 81 values for each metric ($\mathrm{R}^2_{400}, \mathrm{R}^2_{405}, \ldots, \mathrm{R}^2_{800}$). The optimal results of each ML model were also based on the mean of the 81 metric values, given as $\overline{\mathrm{R}^2}$, $\overline{\mathrm{APD}}$ and $\overline{\mathrm{NRMSD}}$.

7.4.1 Atmospheric Correction Results

In this study, the two ML models were used to predict $L_w(\lambda)$ from Rayleigh scattering and absorption corrected radiance ($L_{rac}(\lambda)$), θ, θ_0 and $\Delta\phi$. Before finding the optimal results, a hyperparameter optimization study was done by training the ML models on a range of different Savgol filters and hyperparameters. The ML models giving the best results based on both the Pearson coefficient and NRMSD are presented in Table 7.2. The models are compared further by the metrics mentioned above and execution times.

Table 7.2.: Optimal results when predicting $R_{rs}(\lambda)$ from $L_{rac}(\lambda)$, θ, θ_0 and $\Delta\phi$ using NN and Partial Least Squares Regression (PLSR) with respect to time complexity. Computational time to fit the model ($\mathrm{T_{fit}}$), computational time to predict the output ($\mathrm{T_{pred}}$) and the number of training data ($\mathrm{N_{train}}$) are given.

Metrics	$\overline{\mathrm{R}^2}$	$\overline{\mathrm{APD}}$ [%]	$\overline{\mathrm{NRMSE}}$
NN	0.999	4.42	0.045
PLSR	0.974	34.1	0.197

Time	$\mathrm{T_{fit}}$ [s]	$\mathrm{T_{pred}}$ [s/N]	$\mathrm{N_{train}}$
NN	675	1.3e-3	91702
PLSR	166	1.0e-4	91702

AC of multispectral L_{rac} with a multilayer NN done in [200] produced $\mathrm{R}^2 > 0.993$ for all 7 bands in VIS (412, 443, 488, 531, 547, 667 and 678 nm) and $\overline{\mathrm{APD}}$ = 3.1 %. The NN trained on HSI data showed comparable results, see Table 7.2, with $\mathrm{R}^2 > 0.992$ for all 81 bands and $\overline{\mathrm{APD}}$ = 4.4 %, and $\overline{\mathrm{R}^2}$ calculated from the NN was at 0.999, which was higher than 0.996 reported by [200]. These results imply that both models are able to predict the spectral relationship between L_{rac} and R_{rs} with similar accuracy.

7.4.2 IOP Prediction Results

Several ocean color algorithms [9] can predict IOPs from $R_{rs}(\lambda)$ based on empirical relationships derived from $in\text{-}situ$ measurements, like the non-linear OCx algorithm [16, 209]. Here, NN and PLS models were trained to predict [Chl a], $a_{cdom}(443)$ and TSM from $R_{rs}(\lambda)$ from the simulated HSI $R_{rs}(\lambda)$ data. Different Savgol filters were applied to the input data for each NN model. The best results using NN to predict IOPs are shown in Table 7.3.

Table 7.3.: Predicted chlorophyll concentration ([Chl a]), $a_{cdom}(443)$, mineral concentration (TSM) from $R_{rs}(\lambda)$ with NN and PLSR validated with R^2, APD, and NRMSD.

	[Chl a]		$a_{cdom}(443)$		TSM	
Metrics	NN	PLSR	NN	PLSR	NN	PLSR
$\overline{\mathrm{R}^2}$	0.999	0.987	0.996	0.961	0.998	0.994
$\overline{\mathrm{APD}}$ [%]	8.84	127.1	16.8	84.5	42.4	65.0
$\overline{\mathrm{NRMSE}}$	0.0353	0.271	0.0933	0.275	0.108	0.178

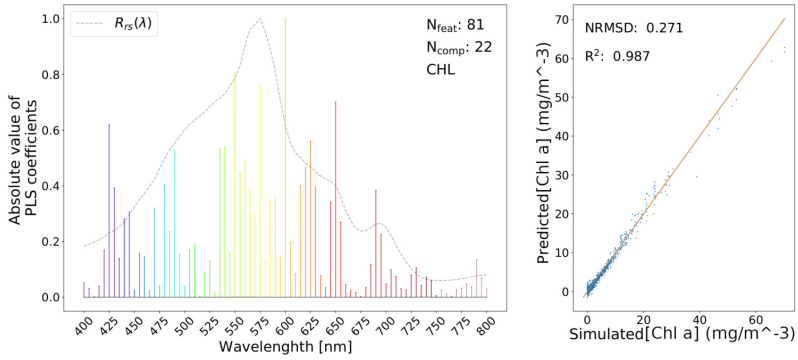


Figure 7.3.: Normalized absolute values of PLS coefficients as a function of wavelength bands for predicting [Chl a] together with scatterplots of the predicted and simulated [Chl a]. Number of features ($\mathrm{N_{feat}}$) and number of components ($\mathrm{N_{comp}}$) are highlighted in the left plot. The grey dashed curve represent one simulated $R_{rs}(\lambda)$. Illustration by Ole Martin Borge.

7.5 Discussion and Conclusions

AccuRT was used to simulate HSI data representative for challenging coastal waters, which could be used to train ML models. When predicting R_{rs} from ToA radiance corrected for Rayleigh and absorption (L_{rac}), all ML models resulted in $\mathrm{R}^2 > 0.968$, indicating that they were able to predict the spectral relationship between L_{rac} and R_{rs}. The best results were obtained with the NN algorithm (R^2=0.999), especially compared to the linear PLS model (R^2=0.974). However, the PLS provided interpretable coefficients, as can be seen Figure 7.3, and shorter prediction time. A specific application or set of constraints will determine the most applicable model.

Unlike many standard AC algorithms, these models seem to be capable of doing AC without short-wave infrared (SWIR) bands, as they were trained on HSI data in the wavelength region $400\text{-}800\,\mathrm{nm}$. The higher spectral resolution in the NIR bands can be a possible explanation as to how the HSI based ML models can be able to estimate and compensate for different atmospheric contributions to the measured signal. Finally, the NN approach could also be used for water IOP prediction, and provided $\mathrm{R}^2 > 0.999$ when predicting [Chl a] from R_{rs}. The results when using synthetic data are comparable or outperform the results reported in [200] on simulated data. It should be noted that the underlying simulated data was trained for multispectral data in [200], and not HSI data as in this chapter.

The different AC algorithms based on ML after training are not computationally complex and, as shown in Table 7.3, can be executed quickly, and therefore could suit operational use in satellites as a part of the on-board data processing pipeline. However, these results do not account for the aforementioned pre-processing. For future research, the ML models should be tested on $in\text{-}situ$ data and be validated against more conventional AC algorithms, such as FLAASH, ATREM or POLDER [4, 9].

Acknowledgements Patrick J. Espy, Professor at NTNU Department of Physics, provided supervision, comments and guidance.

Chapter 8

An Approach for Hyperspectral Chlorophyll-a Concentration Estimation

We cannot cheat on DNA. We cannot get round photosynthesis. We cannot say I am not going to give a damn about phytoplankton. All these tiny mechanisms provide the preconditions of our planetary life. To say we do not care is to say in the most literal sense that "we choose death."

Barbara Ward, Baroness Jackson of Lodsworth

Hyperspectral data processing for monitoring phytoplankton biomass dynamics in time and space has become more viable with new types of instrumentation, such as the one to be deployed on the HYPSO-1 satellite. Methods used operationally for retrieval of [Chl a], have traditionally been developed for multispectral systems and optically deep waters.

Coastal waters are essential for the aquaculture industry, and a higher spectral resolution can provide greater insight and better predictive power. However, data rate limitations make using sensors with a high spectral resolution, such as hyperspectral imagers, more challenging. In this chapter, estimation of [Chl a] from top of the atmosphere reflectance using “Partial Least Squares”- and “Least Absolute Shrinkage and Selection Operator”-regression is compared with the *internal consistency* of the global OC4 algorithm by NASA Ocean Biology Processing Group after AC.

The models perform better in terms of NRMSE and R^2 when validated with a subset of the total data and with different scene geometry. This is demonstrated by using data from the Hyperspectral Imager for the Coastal Ocean (HICO) mission, processed through SeaDAS 7.2. The standard processing forHICOdata provided by SeaDAS 7.2 is used for AC.

With traditional band-ratio algorithms for [Chl a] estimation, like OC4, it has been shown that additional spectral information can improve the accuracy [16, 210]. Successful application of band-ratio algorithms in optically complex waters is often challenging due to overlapping of spectral signals from phytoplankton in the form of [Chl a], CDOM, and TSM that confound the model [211].

Nonlinear machine learning methods, e.g., NN or kernel-based regression models such as Gaussian Process Regression (GPR) and Support Vector Machines (SVM), can give good [Chl a] estimations but will also have a higher level of complexity [212]. The relative relevance of the input features is less transparent, and it can be more challenging to foresee the model behavior. The models themselves will be computationally more demanding to use for estimation when compared to linear prediction models [143, 211, 212]. However, for any linear model, nonlinear patterns can be compensated for by the appropriate pre-processing or kernel functions [213].

PLS regression has been shown to perform with greater accuracy than optimized band ratio algorithms when predicting [Chl a] with field-retrieved hyperspectral water-leaving reflectance [211]. Hyperspectral water-leaving reflectance values are dependent on an accurate correction of atmospheric effects [9, 16]. Proper compensation for the atmospheric effects can be challenging to achieve in coastal regions, and hyperspectral AC for ocean color is an active area of research [9, 14, 16].

This chapter shows two approaches to perform regression analysis and model development using top-of-the-atmosphere pseudo-reflectance values from the hyperspectral image data from theHICOmission directly to estimate the [Chl a]. " ... historically, the atmosphere has been treated as a preliminary step to ocean techniques, but advanced methods for coupled ocean-atmosphere retrievals are an area of growth in the future." [1] The approach presented here thus compares this *area of growth* with a more traditional method for [Chl a] estimation. The [Chl a] concentration data used is derived from processingHICOdata using SeaDAS 7.2. Then the results from the regression are compared to the OC4 band-ratio algorithm, which is a commonly used parameterization when estimating [Chl a] from ocean color [16].

With the approaches, as they are presented in this chapter, the PLS models developed are intuitively interpretive, less computationally demanding, and generate promising results when relying on the two designs for validation.

8.1 HICO data and SeaDAS

The Hyperspectral Imager for the Coastal Ocean (HICO) mission was a hyperspectral instrument onboard the International Space Station capable of capturing scenes with 128 different wavelengths in a range from 350 to 1080 nm at a 5 nm resolution [127].

Here, hyperspectral scenes from theHICOmission, processed and quality controlled through NASA OBPG software SeaDAS 7.2, are used as the ground truth values of [Chl a]. The reflectance

data used here s derived from the standard atmospheric co ecti n pro ed by SeaDAS 7.2 for HICO.

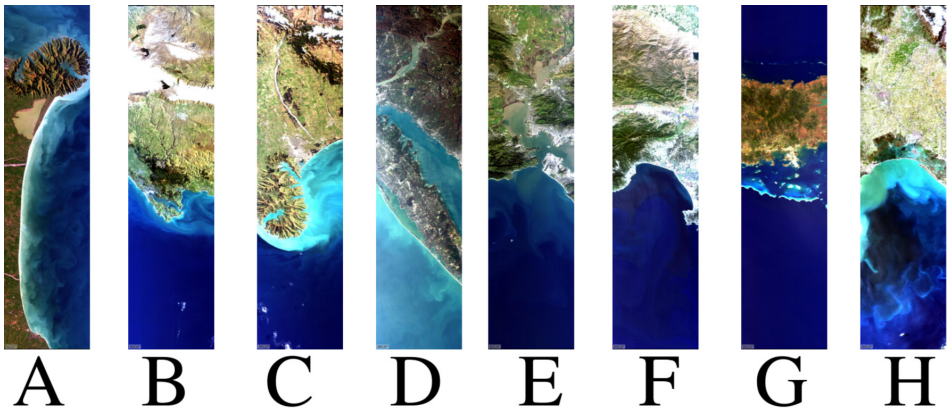


Figure 8.1.: HICO Sample Image Gallery Scenes from different locations around the world used for training and validation.

The scenes were selected due to the minimal adverse effects from atmospheric interference and good overall imaging quality from theHICOSample Image Gallery [214]. The scenes can be seen in Figure 8.1, and their locations are, from the left, southeast coast of New Zealand (A-C), US west coast (D-F), New Caledonia, and Italy.

The [Chl a] concentrations in mg/m^{-3} for the training and the verification data sets are displayed in Figure 8.2. TheHICO[Chl a] data used as ground truth are derived through SeaDAS, which, as standard, used the OC4 algorithm with MERIS coefficients and wavelengths [209]. In SeaDAS, the hyperspectral data is subsampled to the MERIS bandwidths when deriving [Chl a]. From each scene, 700 sample spectra are taken for training, and 300 different spectra are taken for verification in the initial validation procedure. In the secondary validation procedure, the fourth scene from the left, scene D, in Figure 8.1 is kept out and only used as a validation data set. This scene was selected as it has close to the full dynamic range of the [Chl a] considered here. See Section 8.3 for details about the validation procedures. The [Chl a] concentration for each sample can be seen in Figure 8.2 for the first validation procedure.

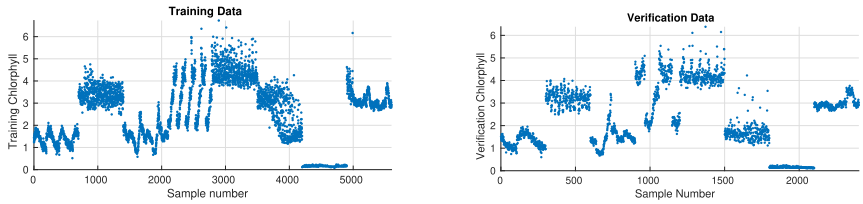


Figure 8.2.: [Chl a] concentrations for the sampled data

8.2 Methods

This section presents a conceptual description of the different algorithms and their advantages and shortcomings. A complete derivation of the algorithms is beyond the scope of this chapter, but references are provided.

The pre-processing of the input variables in this chapter incorporates known physical relationships [143] to simplify the relationships that the different linear regression models are trying to find. Only the radiance signals from 400 to 900 nm are used; only the 87 spectral bands that are recommended to use are used. The wavelengths outside this range have been reported as noisy [127]. No dimensionality reduction or other methods to reduce the number of variables as part of pre-processing have been performed.

ToA reflectance $\rho(\lambda)$ is defined here in the following way; at a given wavelength λ, to be related to the radiance $L(\lambda)$, the extraterrestrial solar irradiance $F_0(\lambda)$, and the solar-zenith angle θ_0, as given in equation (8.1) [9].

$$\rho(\lambda) = \pi L(\lambda)/(F_0(\lambda)\cos(\theta_0)) \tag{8.1}$$

First, with the assumption that the variations in $F_0(\lambda)$ are minor compared to the sun angle effects, the reflectance values $\hat{\rho}(\lambda)$ are approximated by dividing the radiance data with the solar zenith angle.

Secondly, the effect of attenuation of light is accounted for by computing the log values of the approximated reflectances as $\tilde{\rho}(\lambda) = \log_{10}(\hat{\rho}(\lambda))$ [215].

Finally, the input variables have been centered and scaled to adhere to different absolute values and variation for a given wavelength [143]. This is given in equation (8.2), where $\hat{x}$ is the new variable, $\tilde{\rho}(\lambda)$ is the old, $\bar{\rho}(\lambda)$ is the mean, and σ_ρ is the standard deviation.

$$\hat{x} = \frac{\tilde{\rho}(\lambda) - \bar{\rho}(\lambda)}{\sigma_\rho} \tag{8.2}$$

8.2.1 Global OC4 Algorithm by NASA OBPG

The OC4 algorithm developed by NASA OBPG [16, 209], presented in equation (8.3), returns the near-surface concentration of [Chl a] in mg/m^{-3}. The algorithm uses an empirical relationship derived from in situ measurements of [Chl a] concentration and corresponding above-water remote sensing reflectances R_{rs}, with four spectral bands [16]. In this chapter, the implementation of the OC4 algorithm uses the spectral bands closest to the ones used by the SeaWiFS multispectral imager [16, 216]. $R_{rs}(\lambda_{green})$ is the band closest to 555 nm, and $R_{rs}(\lambda_{blue})$ is the maximum of the bands closest to 443, 490, or 510 nm. The a_i coefficients used in the implementation of OC4

presented here were determined using ordinary least squares on the training data presented in Figure 8.2 [143]. The equation is given as

$$log_{10}(chl\_a) = \sum_{i=0}^{4} a_i \left(log_{10} \left(\frac{R_{rs}(\lambda_{blue})}{R_{rs}(\lambda_{green})} \right) \right)^i . \tag{8.3}$$

8.2.2 Partial Least Squares Regression

PLS regression iteratively relates data matrices using linear multivariate models that reduce collinearity and noise within a given dataset. It is a two-step algorithm that first finds uncorrelated components in the variables of a given data set and then performs the least squares regression on these components. A more in-depth description of the algorithm can be found in [217]. This approach generates models with high levels of interpretability, but without appropriate pre-processing, they cannot accommodate for strong nonlinear effects [143].

8.2.3 Least Absolute Shrinkage and Selection Operator Regression

Least Absolute Shrinkage and Selection Operator (LASSO) regression [218] was performed using the approach found in equation (8.4), where y_i is [Chl a] value, β_i is the regression coefficients, $\mathbf{X}$ is a matrix with all the pre-processed data, and t is a threshold that was iterated over with 10-fold cross-validation of the training data. The threshold that gave the lowest mean square error was selected.

This also generates models with high levels of interpretability, but in their simplest form, cannot accommodate significant nonlinear effects. This method does not seek to find any covariation between variables. In Eq. 8.4, a vector of all ones is represented as $\mathbf{1}_N$.

$$\min_{\beta_0,\beta} \{ \|(\mathbf{y}_i - \beta_0 \mathbf{1}_N - \mathbf{X}\beta\|_2^2 \} \qquad s.t. \quad \|\beta\|_1 \leq t \tag{8.4}$$

8.3 Results & Discussion

Presented in this section is a discussion comparing the results in terms of regression coefficients, Normalized Root Mean Square (NRMSE), and R-squared (R^2) values for OC4, PLS and LASSO regression.

All models were tested with the verification data set shown in Figure 8.2, as well as separating out the fourth scene from the left in Figure 8.1, scene D as the scene inhibits the full dynamic range in terms of [Chl a] investigated here.

8.3.1 OC4 Algorithm

The a_i coefficients used in equation (8.3) for the OC4 is computed by taking the least-squares fit of the training data set. This should give OC4 the best possible starting point.

As can be seen in Figure 8.4 the OC4 algorithm performs similar to what has been previously reported in the literature [16, 209]. No clear trend in the residuals can be found with validation using a subset of the total data, i.e., the verification data set. The algorithm struggles to determine [Chl a] $> 4mg/m^3$, with the coefficients found through ordinary least square fitting.

For the validation with a separated scene, scene D in Figure 8.1, the OC4 algorithm can determine a relative [Chl a] concentration within a given scene from SeaDAS 7.2, but is not able to quantify it accurately. This is as expected according to to [16].

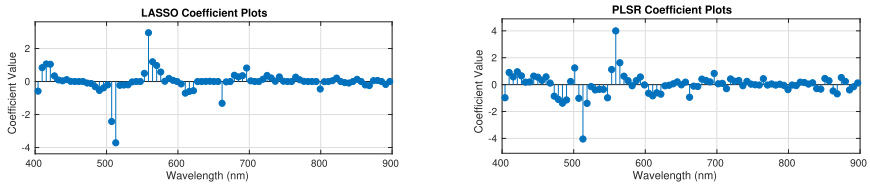


Figure 8.3.: Regression coefficients from PLSR and LASSO

8.3.2 Regression Models

LASSO and PLS regression are both linear regression models that provide interpretable coefficients, as can be seen in Figure 8.3. The models developed deployed 10-fold cross-validation with the training data set when determining the regression weights. For the PLS regression, the average mean square error from the 10-fold CV was used to select the number of components, i.e., 20 components, to be used in the regression.

As can be seen in Figure 8.4, both the LASSO and PLS regression models perform similarly on the given data set. From the results, PLS regression has a potentially negligible higher performance in terms of NRMSE and R-squared when compared with LASSO. It should be noted that LASSO here uses 67 of the original 87 variables, which can be valuable from an operational point of view in terms of execution time. The regression models also struggle to determine higher [Chl a]. Possibly, due to lack of data representing that part of the data or nonlinear effects.

For the validation with a separated scene, scene D in Figure 8.1, the regression models are also able to determine a relative and more accurate quantification of the [Chl a]. The regression models have a better performance than the OC4 algorithm with the validation scheme using data from all scenes.

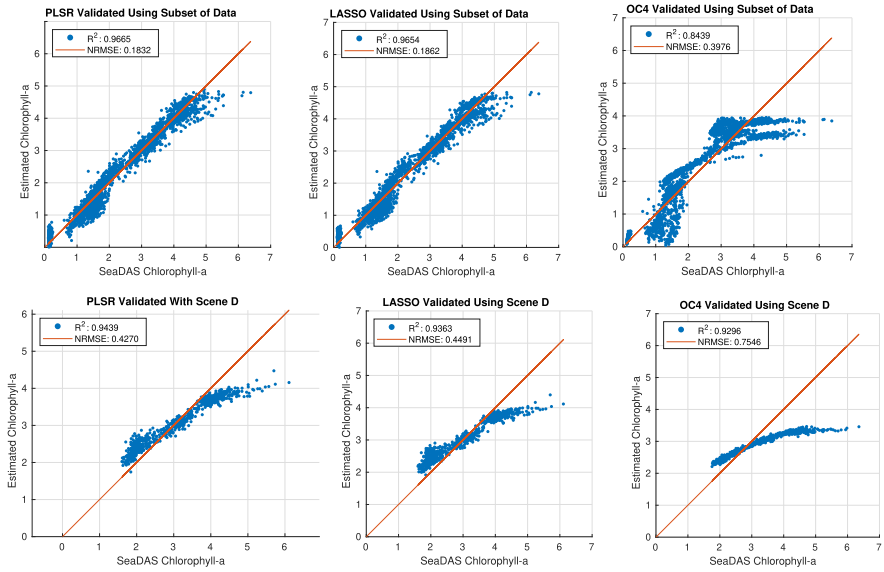


Figure 8.4.: Results from scenes showing Normalized Root Mean Square and R squared.

8.3.3 Comparison

Please remember that this is a test of the *internal consistency* of OC4 within the SeaDAS 7.2 software, i.e., how different bands used as a basis for OC4 will perform. A more correct data set with ground-truth samples measured by other means would be a better study than what is given in this chapter. With the method used here for comparison, it is fair to assume that the performance of the OC4 algorithm would be affected by how the data is prepared.

With the preprocessing described in section 8.2 the variables need only be atmospherically corrected and moved into a 4. order polynomial kernel [213] away from having the same form as equation (8.3). The data representation chosen for regression incorporates a well-characterized nonlinear physical relationship, e.g., the transformation from radiance to reflectance and light attenuation. This aims to ensure that the ML algorithms, i.e. different forms for regression, do not put much emphasis on estimating these nonlinear relationships.

As can be seen in Figure 8.4, both the LASSO and PLS regression models perform better than the OC4 algorithm in terms of NRMSE and R-squared for the validation schemes used here. The two regression models have a similar performance in terms of the chosen metrics, but the execution time of the LASSO regression was, on average, 1.8 times faster in the given MATLAB implementation. It is expected that the LASSO model would execute faster on the target hardware of HYPSO-1 as well.

From the coefficient illustrated in Figure 8.3, it is clear that the LASSO and PLS regression approach emphasizes similar parts of the electromagnetic spectrum. It is also clear that some of the coefficients, $>$ 555 nm and $<$ 443 nm, have high expressive power in determining the total [Chl a] concentration. The wavelengths 555 nm and 443nm indicate the maximum and minimum of the OC4 algorithm. That additional spectral information improves chlorophyll determination, and this corresponds well with other findings investigating band-ratio algorithms [16, 210].

8.4 Conclusions

The presented ML models may provide absolute measurements of [Chl a] concentration from only using the measured ToA radiance and the attitude and solar angle information related to the hyperspectral sensor.

Multivariate methods such as PLS seem to be suitable for deriving some bio-geophysical variables of interest, such as [Chl a]. At the same time, these linear methods can provide an interpretable derivation of results in the form of coefficients. This explainability makes it easier to understand why the models derive the values that they do, which can reassure the end-user.

The LASSO model, when compared to PLS, provided a reduction in the average computational time per pixel, and this encourages its use in computationally constrained systems. These results are implementation and hardware dependent, and considerations related to this is not accounted for in work presented here.

When doing ML there is a considerable advantage of having large data sets with verified ground truth, but this is not widely available for hyperspectral ocean color remote sensing data; thus,HICOdata quality is assured through SeaDAS 7.2 was used. When more ground truth data become available with future space missions and systems, better models could be developed using this approach. Also, the data used in this chapter only represents a subset of the full range of naturally occurring chlorophyll-a concentrations. Again, with more data, better models could be developed.

Preprocessing with targeted binning of spectral regions of interest for chlorophyll-a concentration, found through analysis of the regression coefficients, could improve the signal to noise ratio of the spectra and in return, improve the estimation performance, and could be a topic for further investigation when HYPSO-1 can provide data.

Chapter 9

Digital Engineering Development for HYPSO-1

... plans are useless, but planning is indispensable.

Dwight D. Eisenhower

Digital engineering is increasingly introduced for managing and supporting the development of systems for space. However, few academic teams have the competency needed to manage projects using digital engineering and systems engineering. The subject of this chapter is the Academic CubeSat project, HYPSO-1, where a variety of digital engineering techniques are used to manage the project.

Here the tailoring that has been applied to fit the academic environment including students from different disciplines and levels of maturity is described. This chapter show how our customized Scrum methodology for hardware and software integrated with a workflow in a digital tool environment has given positive results for both the team and the system development.

This chapter also discusses how the HYPSO mission introduce new members to the team and how to train them to work with digital engineering as a multi-disciplinary team. This chapter present how the systems engineering and project management activities have been integrated into the academic CubeSat project, evaluate how well this fusion worked, and estimate its potential to be used as a guide for other digital engineering projects.

The digital transformation that is taking place in all elements of society calls for continuously updated knowledge for leaders and engineers. The increasing project complexity introduced by the advent of embedded systems and Cyber-Physical Systems (CPS), and the tools needed for developing them, challenges managers to re-think the approach to leading projects and people to ensure knowledge management and project success [219]. While this is challenging in industrial settings with experienced engineers and support systems, developing complex systems in an

academic environment adds factors such as high turnover, coursework, lack of multidisciplinary teamwork experience, and fewer competent Systems Engineering (SE) and Project Management (PM) resources.

Digital engineering and Model-Based Systems Engineering (MBSE) are proposed as tools to manage the challenges of developing systems, delivering integrated multidisciplinary product development from concept through the product life-cycle to retirement. We adopt the Digital Engineering (DE) definition of the U.S. Department of Defense (DoD): "an integrated digital approach that uses authoritative sources of system data and models and a continuum across disciplines to support life-cycle activities from concept through disposal" [220, p. 340]. For MBSE, we use the definition provided by International Council On Systems Engineering (INCOSE): "The formalized application of modeling to support system requirements, design, analysis, verification, and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [221]. However, choosing the approach tools and methods to introduce and adopt DE is equally challenging and requires both human and technical resources.

Concurrent with the advent of digital engineering, approaches such as Scrum and Extreme Programming (XP) have increased in popularity both for hardware and software [222]. The Scrum methodology allows for agile product development to respond to changing demands from stakeholders and new technology developments while continuously delivering features. The digital Scrum tools also provide a system that supports *project management* through feature and schedule management, *product management* through scope and verification management, and may be integrated with the digital design artifacts. Extreme Programming takes iterative development to an "extreme" level, with short iterations, continuous test development, pair programming, continuous integration, and frequent releases [223]. In software projects where there is scientific code development, and requirements are either unknown at the beginning or frequently change, XP or Scrum are suitable over other traditional approaches [224].

Students in academic projects face the challenge of balancing coursework and project work. The students follow the school year, so long-term academic projects must adapt their expectations to this fluctuation, and there is a high natural turnover in the team composition when students graduate. Academic projects may have fewer resources and fewer support systems that product development often necessitates (e.g., a procurement department or quality assurance knowledge) [225, 226]. The university context requires attention to knowledge transfer and management, and digital engineering is a tool that can be applied and must be managed to enable a good development environment.

This chapter is based on the longitudinal case study of an academic CubeSat where the students typically join in September and leave in June the following year, although some students join in January and leave in June the same year. They contribute to the development of the CubeSat through work toward a thesis in either software, hardware, or theoretical studies. We explore the development cycle of a CubeSat in an academic environment using digital engineering tools and describe how they have been tailored. Furthermore, we discuss how MBSE has been applied and

what barriers for use were experienced. We found that using agile practices powered with DE tools and processes greatly improved information sharing and knowledge management and that the introduction of remotely accessible Hardware-In-the-Loop (HIL) setups coupled with a defined workflow has enabled improved verification, validation, and integration activities.

9.1 Background

9.1.1 Agile Methodology and Development Practices

Using agile methodologies in software and hardware development has gained popularity in the past decades, focusing on continuous feedback from the customer and the ability to react to a changing environment [227, 228]. The word "agile" has its etymological source from the Latin word *agilis*, which means "can be moved easily, light", and from the French word *agere*, which means "to drive, to be in motion" [229]. In software development, the agile methodology gained popularity in the late 1990s, and "Manifesto for Agile Software Development" [230] with its 12 guiding principles was published in 2001. The manifesto includes principles that focus on delivering the highest value to the customer, to allow for changing requirements, frequent and iterative deliveries of software, motivating individuals, face-to-face conversations, measuring progress through working software products, simplicity, reflexive practices, and believing that the best designs come from self-organizing teams [230].

At universities, software and hardware development serve both to assist scientists in gathering data and teaching technology and product development to engineering students. In most cases, the development is not done to deliver a mass-produced product or service, but to contribute to new knowledge and research. A key challenge of scientific software development is that the scientists often have formal education in a field other than computer science, for example, in biology, remote sensing, electronics, or radio technology, but need custom software to address their discipline-specific research questions [231, 232]. Given the open-ended nature of research projects, the process of requirements specification lacks maturity compared to industrial development projects, making it challenging to plan the development and test the software. Furthermore, the scientific software development does not "stop" when the first research project ends, but it may be reused in a different research project with different goals, and new scientists desiring new functionality [233].

Best practices for scientific software development include: write programs so that the other researchers understand and stick to a code style and formatting, make the frequently used commands easily accessible, incremental development with continuous testing, use version control, "plan for mistakes" and use unit testing, improve performance after the functionality is there, document the design and interfaces, and choices made during development, and collaborate on code development and do code reviews [231]. Typical challenges facing scientific software teams are "compromising between feature demands and quality control; code ownership and management during evolution; data organization and curation; and quality assurance of heterogeneous components, (...) and a tendency for prototyping practices to be employed even when scientific software intended to be

used in production was being written [234, p. 47:6-7]." In Arvanitou et al., software practices for scientific development were discussed based on an extensive literature study [232]. They found that most scientific software engineering literature has studied process improvement, ease of development, testing and verification, project management, coding, and quality assurance. Furthermore, that *performance*, *maintainability* and *development productivity* were the highest priorities for the scientists.

In a survey of agile methods in scientific programming in disciplines such as bioinformatics, climate scientists, and aerospace, it was found that the agile method XP has been applied successfully in projects where requirements and design cannot be known in the planning phase of a project [234]. Furthermore, agile practices such as iterative development, continuous integration, and version control were prominent. In contrast to commercial and industrial software development, no declared or identified customer reviews the software features. However, scientific publications can be analogous to customers in which the scientists receive feedback on what they have developed [234, 235]. Sletholt et al. [233] conducted a literature review against 35 agile practices from Scrum and XP, and found some support that agile practices are suited to testing-related activities.

Agile practices in teaching have gained popularity since the 2000s [236, 237], where Scrum or XP have been the most prominent methods, and typically found in either software or capstone projects. The students benefit from learning hands-on project experience, learning to prioritize work tasks, gaining communication skills, and providing and receiving assessment on work done openly. However, there may be challenges in terms of balancing time commitments, for example, having concurrent development sessions, or tailoring the Scrum processes to suit the different needs of team members [237]. Lundqvist et al. [238] reported on teaching agile in cooperation with industry. They highlighted the importance of ownership, customer engagement, also called the industrial partner, and the allocation of academic resources to support the academic teams.

According to a study from Australia in 2015, employers want both technical skills and non-technical professional skills such as "being able to communicate effectively," "ability to organise work and manage time effectively," "being willing to face and learn from errors and listen openly to feedback," "being able to empathise with and work productively with people from a wide range of backgrounds" [239, pp. 263–264]. A similar study conducted in Norway also highlighted these points [240]. However, the traditional form of classroom teaching may not effectively facilitate the development of these skills. Using CubeSats for training students in cross-disciplinary projects has been studied and discussed [22, 225, 241–243]. Some principles for agile SE that have been suggested include (1) focus on delivering customer value, (2) team ownership, (3) embrace change, (4) continuous integration, (5) test-driven, and (6) taking a scientific approach to systems' thinking [235, 244]. Many of these principles are aligned with transferable skills students can be expected to have when they graduate [239, 240].

9.1.2 Digital Engineering

Digital engineering goes beyond "just" using computer tools to aid engineering but includes the engineering process and approach to development. Choosing a DE strategy should be done based on the resources available and the needs of the organization. A framework that assesses the DE competence was developed by the Space Engineering Research Center (SERC) which looked at the following areas: *adoption*, *velocity/agility*, *knowledge transfer*, *user interface*, and *quality* [245]. While the framework did not specify how to measure competence in each area, it listed different factors and examples of processes or outcome metrics that could be used. Some factors identified can be categorized as objectives for why DE measures are incorporated, others as factors which may influence the adoption, and other factors as outcomes and direct competencies the organization can gain with DE practices. DE has a strong relationship with MBSE and Model-Centric Engineering (MCE), and establishing a "single source of truth" for a project [220]. However, there is currently no single solution for the whole system lifecycle to provide an authoritative source of truth. Most work-forces and organizations need to transition their methods and methodologies to DE and incorporate it into their engineering practices, and ensure possibilities for collaboration and information sharing throughout the system lifecycle between developers and the stakeholders. Most university CubeSat teams use some degree of DE, such as employing version-controlled software repositories, using CAD tools, shared cloud documentation, and using cloud-based issue tracking or project management tools to achieve integration in the management of knowledge [22].

Garzaniti et al. [246] also describe the use of Scrum using an online tool to manage the work in an academic CubeSat team. The results presented were from the preliminary design phase of the space hardware. Garzaniti et al. [246] found that the Scrum approach helped react to unforeseen changes and delays, even when the changes impacted external manufacturers. Furthermore, that it takes time for the team to become accustomed to Scrum and the scoring of issues, similar to [237]. Huang et al. [247] describe the development of a CubeSat using agile practices. They highlight the importance of tailoring the approach to the project's needs, using interactive design reviews to produce as much feedback as possible, empowering smaller teams to enable faster decision-making and ownership, and allowing for continuous testing and improvement.

9.2 The HYPSO Case Study

9.2.1 The HYPSO CubeSat Project

In this chapter, we report on the case study of the CubeSat project HYPSO. It is the first research CubeSat mission for the NTNU, as a part of a strategy of monitoring coastal areas using autonomous assets [23]. The project's mission is to:

> *"To provide and support ocean color mapping through a Hyperspectral Imager (HSI) payload, autonomously processed data, and on-demand autonomous communications in a concert of robotic agents at the Norwegian coast."*

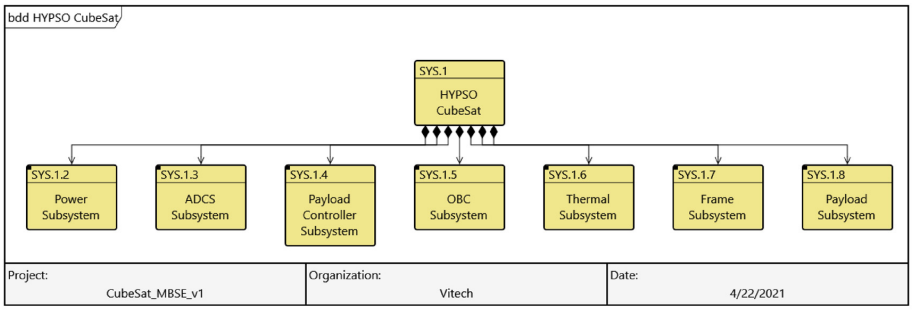


Figure 9.1.: Overview of the HYPSO CubeSat and its subsystems. Model made using CORE/GENESYS. Illustration By Evelyn Honoré-Livermore.

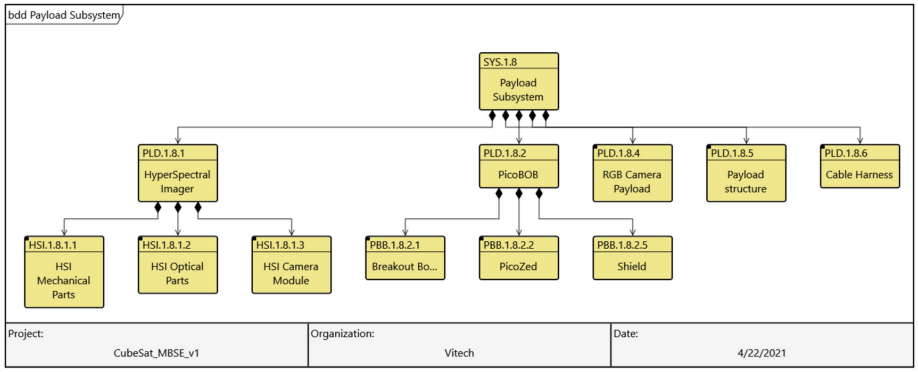


Figure 9.2.: Overview of the payload developed by the HYPSO team. Model made using CORE/GENESYS. Illustration By Evelyn Honoré-Livermore.

The university CubeSat team develops the payload, which consists of an optical telescope, a COTS camera unit, a COTS processing unit, an electronics interface board, an electrical harness, software to control the payload and to perform the image processing, and mechanical support structure which also acts as the mechanical interface to the satellite bus. Block diagrams of the spacecraft and the payload are given in Figure 9.1 and Figure 9.2, respectively. Apart from the above-mentioned COTS components, all have been developed in-house. In addition to the payload, there is also the development of a local ground station and the mission operations center and associated procedures and functionality, effectively resulting in a System of Systems (SoS).

The CubeSat project team includes 10–20 MSc and BSc students, one electronics engineer, a procurement officer, 6–8 PhD/Post.Doc. researchers, and professors that are supervising the thesis work or offer experience and support. The project manager is a Ph.D. candidate examining the value of MBSE to deliver the CubeSat on time and within schedule. The researchers typically join the project for 2–4 years, and the students for 4 (BSc) or 9 (MSc) months when they write their thesis.

The backgrounds of the students vary, but typically they are enrolled in engineering cybernetics, embedded systems, electronic systems, product development, or material science. Some of the students have experience working in teams and sometimes multidisciplinary development through previous coursework or volunteer organization. However, not many have experience with product development, which typically has more unknowns than course-organized project work.

The project had its first major milestone in December 2017, the Mission Design Review (MDR). Before this, there had been some software development, mainly focused on algorithm development for processing, without target hardware or system in mind. The overall system maturity timeline is shown in Figure 9.3, and a more detailed timeline of the progress in 2020 is shown in Figure 9.4. Most of the integration and HIL testing occurred in 2020.

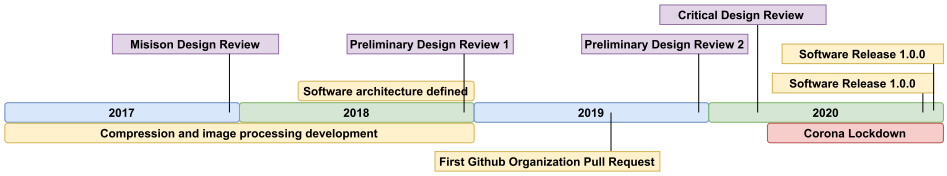


Figure 9.3.: Overall timeline of in-house developed product maturity, including both hardware and software.

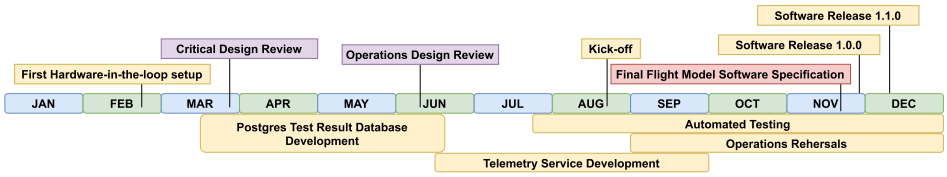


Figure 9.4.: Timeline of product maturity through 2020. A different rendition is given in Figure 11.1

9.2.2 Software System Architecture

The high-level system architecture is given in Figure 9.5, where the flow of signals and data is bi-directional. Some of the items in the software architecture are developed in-house, while others are delivered by suppliers, or interfaced as a service. The architecture was unclear at the beginning of the project, and has been gradually defined throughout the system development lifecycle. The components have also undergone continuous development and updates to the interfaces to a certain degree. The reasons for continuous development and changes are

- New functionality requirements and new performance requirements.
- The inherent constraints of the chosen components.
- The learning and discovery process of developing a CubeSat system for the first time.

Modular software components require that interfaces and software architecture are defined. While the initial software architecture was developed in late 2018, not all interfaces between different components were defined. This meant that much work was required to integrate the in-house developed components with the developed software. Furthermore, the interface definition to other spacecraft subsystems had not been considered before 2018, such that the components also needed adaption to enable integration to the satellite bus. The software-based subsystems allow for hardware to host the functionality of several subsystems. For the HYPSO spacecraft (Figure 9.1), the subsystems "SYS1.3 ADCS Subsystem" and "SYS1.5 OBC Subsystem" are both hosted on the same physical component, the Flight Computer (FC). On the payload, the physical OPU hosts the image processing pipeline, the camera control, the payload operating system, and telemetry services for the payload.

In Figure 9.5, each partition is composed of tightly integrated physical and software subsystems, namely, a cyber-physical SoS. The space environment will affect each of the interfaces between the subsystems and the spacecraft's performance, and the software subsystems need to adjust (for example, pointing the spacecraft towards the sun when the battery levels get low) to ensure functionality and performance. Additionally, to develop hardware components, one needs to consider the software, and when developing software components, one needs to consider hardware limitations, such as data transfer speed limitations or processing hardware physical layout. Furthermore, the "Mission Control Software", and "Mission Operations Center" were not available until mid-2020, which led to the discovery of new functionality and software adjustments to facilitate operations of the payload. When the spacecraft is operational and commissioned, the operator will only interact with the first box (the telemetry display and the `hypso-cli` (user interface translating commands to packets used for communicating) or `nanoMCS` interface) and the `OPU-services` on the HYPSO spacecraft, under the expectation that the underlying system functions as expected. Despite the many hardware and software systems between the operator and the spacecraft, they must exchange information correctly and promptly.

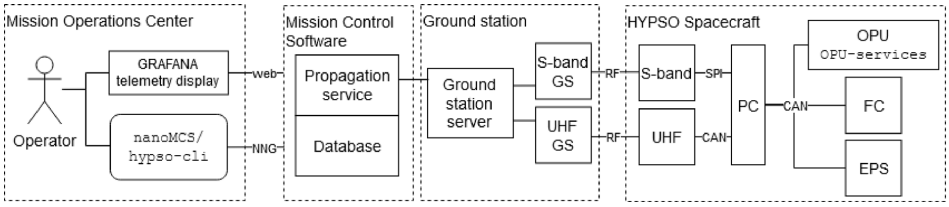


Figure 9.5.: The software system architecture. OPU = On-Board Processing Unit, FC = Flight Computer, EPS = Electrical Power Subsystem, PC = Payload Controller, CAN = Controlled Area Network, GS = Ground Station, RF = Radio Frequency, NNG = nanomsg Next Generation.

9.2.3 Tailoring of the Agile Methodology

The Scrum methodology has been tailored such that the team members deliver either a product increment or a thesis, as shown in Figure 9.6. The sprints typically lasted two weeks, and there was

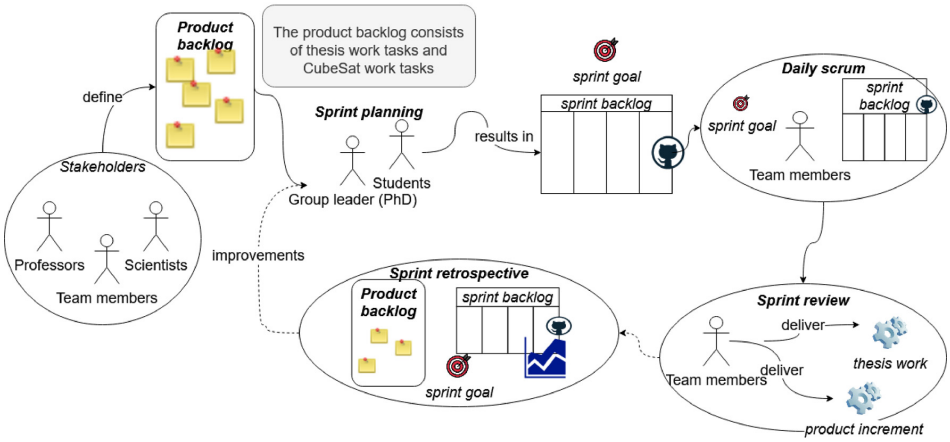


Figure 9.6.: Tailored Scrum process with a product backlog consisting of both thesis tasks and project work tasks. Illustration By Evelyn Honoré-Livermore.

a daily scrum meeting (a stand-up) in which issues were raised or discussed for clarification, in addition to general keeping in touch with each other. The team uses GitHub for managing the code repository and schematics, and providing version control and release management [22]. GitHub is a service that provides users of several different backgrounds and development approaches to work together and at the same time have a coherent overview of the current status of the codebase. GitHub has a plugin for managing Scrum with a *kanban* board. Kanban boards, from the Japanese word meaning billboard, are used to visualize and manage workload by providing an overview of work-in-progress, backlogged items, blocked items, done items, and review-in-progress items. A kanban board is based on *pulling* tasks instead of being *pushed*, which enables the students to take control of their workload. At the same time, the Scrum master (called group leader in Figure 9.6) can control which items are included on the board, so the work that gets done is pertinent to the schedule and the product to be delivered.

Planning, workflow, and continuous integration Planning and developing a complex system are not guaranteed to align well with research goals found in academia. Finding synergies and acknowledging what needs to be prioritized can benefit the development of a CubeSat and provide a better foundation to build and expand research activities. While Scrum traditionally has a goal of delivering a pre-defined MVP at the end of a Sprint, this was not the case for HYPSO. In this case study, participants contribute to components ranging from hardware to User Intrerface (UI). Until the first agreed software release at the end of 2020, as shown in Figure 9.4, the sprint backlogs included issues which the team members "wanted to focus on" and had time to work on. There was an agreement between the team members when selecting issues, and there was a continuous focus on working on issues labeled as "bugs" or mission-required functionality (defined by the group

leader in conjunction with the project manager) instead of issues categorized as “enhancements” in GitHub. Furthermore, each participant developed modules without defined interfaces between them. This made retaining the value added from different contributions, and especially integration, unnecessarily complicated and time-consuming. A standard workflow was proposed to mitigate these challenges, became a part of the on-boarding procedure, and provided the students with a shared repository.

Some of the contributors only participate in the development for as little as one semester, and there are limitations to how complicated the workflow and the development tools can be. To achieve a convenient workflow, development needs to be coherent, and a multitude of development considerations have to be made clear and followed up to ensure the desired quality of the project and product. Continuous Integration (CI), or the practice of integrating contributions from multiple developers into a software product, is beneficial for collaborative code development [248], and is also promoted in XP practices. A workflow focusing on integration was then proposed, i.e., the GitHub workflow [249]. This workflow states that the main branch shall always be working, and any feature or fix to be included in the code base shall originate from a dedicated branch, i.e., there are no development branches that branch out beyond the main branch. This workflow encourages contributors to frequently merge their code contributions into a central repository for review and testing, as is considered a good practice in software development [249], and often mentioned as an important lesson learned from CubeSat development teams [22].

9.2.4 Verification and Validation Using Hardware-In-The-Loop Setups

Verification and validation are essential to ensure that the product functions as specified (verification) and meets the needs of an end-user (validation). Collectively, these will be referred to as testing. In the HYPSO project, several testing regiments were developed to expand the number of reviewers. The software group leader emphasized that approval of a Pull Request (PR) should be done by reviewers, not necessarily involved in the development of the code. In other words, the contributors were required to describe their changes or additions in such a way that “any” software team member could be able to review them. Even though not every team member can review every change, this motivates the developer to make code modifications in such a way that they are understandable to “any” person responsible for reviewing said changes. To become part of the master branch, at least one other person has to approve the suggested changes. When the code changes are committed to a separate feature branch of the central repository, a team member then builds and tests before being accepted as a valid code base addition. If no adverse effects are detected during the review, the pending PR is then merged into the master branch. This review is the manual testing process and ensures that the newly added feature or fix is tested independently and sufficiently. This testing strategy is also described in chapter 11.

In addition to the manual process, several automatic scripts have been developed to do routine tests of nominal operations of the system. While simplifying testing any proposed changes on the

target hardware, this also provides a platform for other types of testing. Several installations of the system, laid out as closely as possible with the actual satellite, were set up to be interfaced remotely by any team member, namely the HIL setups. HIL setups can be used for verification of functional requirements [250], and if deployed on the target hardware, it can also verify performance requirements. Because university CubeSat projects often have limited funding available, having a full *engineering model* (a replica of the system) of the satellite bus and its subsystems is not always feasible. Instead, using a *FlatSat* (a flat satellite) with subsystems provides many of the same functions at a much lower cost. The satellite bus providers often sell FlatSat services at lower fees because the subsystems that constitute the FlatSat can be shared between different customers, or the subsystems can be development models used by the satellite bus providers themselves.

Two HIL setups were developed to facilitate verification and validation activities, and to improve early integration efforts. The HIL setups are shown in Figure 9.7, and are called *LidSat* (because the systems are mounted in an ESD-box lid), and *pHIL* (payload HIL). Both setups use target hardware for the software subsystems, and have different purposes. The pHIL setup is mainly for testing the payload and its communication interface with the command-line interface, while the LidSat is used to test both the payload software and the integration of the payload to the spacecraft. The pHIL is connected to a workstation that is running a continuous integration server. We have used both Jenkins and GitHub actions for this purpose in the project. To test a branch of the software, the branch is first compiled and initiated on the payload. Then Jenkins runs a set of tests on the target hardware. The outcome of the tests (both whether they pass and their performance) is recorded in a database. The central database allows the developers to see how various branches have performed during the test. The test set includes sending several commands that operators commonly use and ensuring that the correct results are obtained for different parameters. The LidSat has both the Electrical Power Unit (EPS) and payload controller connected via a Controller Area Network (CAN), with an additional connection to the rest of the spacecraft subsystems on a FlatSat in Vilnius through the internet with a CAN-over-internet bridge. These are the primary interfaces for the payload, and as such, the FlatSat replicates integration with the spacecraft.

Furthermore, integration testing has been automated by scripting commands sent from the operator computer to the payload. Scripts have been developed to aid other hardware team members in testing nominal operations when mechanical changes are made, and these scripts are also used in a test-to-failure scheme where the procedures are repeated a set number of times or until failure. A script testing the potential performance alterations was also used on the system and a test of the subsystem communication and integration. These tests are run routinely to uncover unforeseen adverse effects of any proposed code changes.

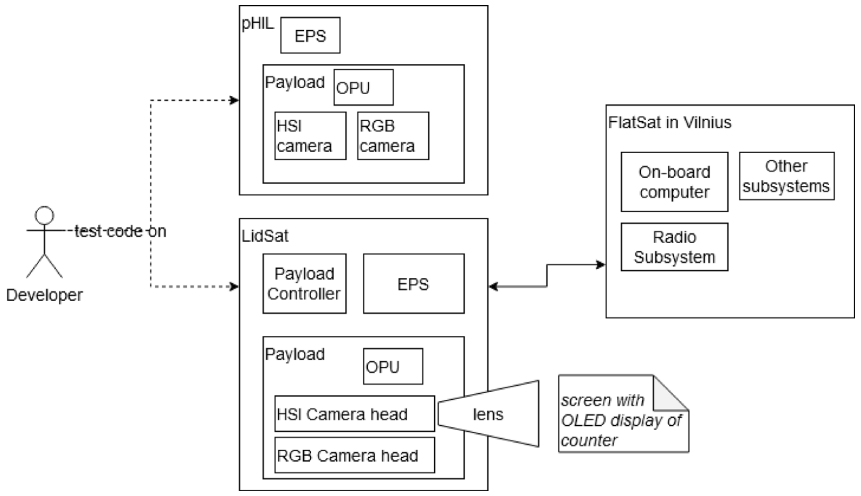


Figure 9.7.: Hardware-in-the-loop test setups.

9.3 Experience Using Digital Engineering in an Academic Project

The product development lifecycle with its DE tools and methods are shown in Figure 12.1. Note that specific tools used for analysis are not shown, as they depend on the specific discipline and task the team member is working on. This lifecycle is supported by the GitHub workflow and the Scrum method for daily work management. Many improvements can be made, but the DE strategy presented here is low-cost and makes use of well-established processes and tools that are readily available. Furthermore, while some training is needed, and there should be an agreement to be consistent, most HYPSO team members agree that the benefits greatly outweigh the cost.

In this section, we will discuss which factors influence the approach to DE, evaluate the effectiveness of using agile practices, the educational aspect of the HYPSO project, and also provide some insights gained during the COVID-19 outbreak and how this relates to DE [226].

9.3.1 Choice of Digital Engineering Strategy

The choice of DE processes for the HYPSO project team was continuously evaluated, with the introduction of new methods and tools as needed. The overall strategy was to adopt and test different DE approaches throughout the project. Typically, the solutions chosen were based on previous knowledge or experience from the team members in other projects. This previous experience also made the training of other team members more straightforward, which is a critical component in adopting new methods and tools.

From the list in McDermott et al. [245], the factors listed in Table 9.1 were chosen. The factors were selected by reviewing the discussions in the project team that led to the DE approach. No quantitative measures of DE competency before and after the introduction of tools were done. However, results from action research have been used as a basis for this chapter.

Table 9.1.: List of factors influencing digital engineering strategy at HYPSO project. Right-hand side shows the sociotechnical factors, while the left-hand side are more technical.

Digital Engineering Competencies			
Category	**Factor**	**Category**	**Factor**
Quality	Traceability	Knowledge transfer	Better information sharing
	System quality		Better information accessibility
	Reduce defects/errors		Improved collaboration
	Improved system design		Better knowledge capture
	Increased effectiveness		Improved architecture
	Strengthened testing	Adoption	General resources for implementation
Velocity/Agility	Improved consistency		Workforce knowledge
	Reduce time		DE processes
	Increased capacity for reuse		Training
	Early V&V		DE tools
	Easy to make changes		Demonstrating benefits
	Higher level of support for integration		People willing to use tools
		User experience	Improved system understanding
			Reduce effort
			Higher level support for automation
			Better decision-making

Adoption The *DE tools* were based on what would have a high adoption rate, be open-source or free/educational license, and that there would be little resistance from the students. For example, the project team conducted polls to decide on which cloud file repository to use, which communication platform to use, and which video conferencing tool to use. This means choosing tools with a well designed user interface, or tools that have been used in other courses, closely linked to *Workforce knowledge*, to reduce the need for *Training* as there are little *General resources for implementation.* The implementation efforts mainly have to be performed by students or group leaders (Ph.D. candidates). The *DE processes* were selected based on recommendations in literature review [230, 249, 251] and recommendations from other CubeSat teams at informal discussions at conferences such as the International Astronautical Conference or Small Satellite Conference. Considerations were made to find processes that would not require too much *General resources for implementation* and that would quickly *Demonstrate benefits* to the project team, to ensure that the team members were *Willing to use tools*.

Knowledge transfer During the first year of the HYPSO project, challenges with *Information sharing* frequently occurred, such as missed hardware changes that influenced both software and hardware performance but were not communicated clearly. Furthermore, the complexity of the system necessitates *Better information accessibility* and *Better knowledge capture*, which were two of the main objectives to fulfill for the DE tools and processes chosen. The Agile methodology used in both hardware and software teams *Improved collaboration* and *Information sharing*. This improvement was achieved by having the issues documented in GitHub and through the daily joint

stand-up meetings. In addition to the technical benefits of using the GitHub workflow, having a standard workflow could also increase the feeling of team cohesiveness and shared understanding of how the fragments can work and should work together through, for example, testing each other's code. The joint stand-up meetings enabled a better understanding of how hardware and testing worked for the software developers and limitations in, for example, physical interfaces, from the perspective of hardware developers. On the other hand, the hardware developers got a better understanding of how the system would be used operationally, and could align their development and prototyping schedule to accommodate for verification and validation activities.

User experience Because the DE strategy involved stand-up meetings, 3D-printed hardware prototypes, and HIL test setups, team members acquired an *Improved system understanding*. While it is difficult to prove an improvement, discussions during review meetings have been less about clarification and more about design enhancements and future development. The first iteration of the agile methodology used a physical kanban board, which was not adopted well by the team. Introducing a GitHub kanban board *Reduced the effort* needed to separate software code development from the process of managing the development. This separation is a clear advantage of using DE tools and processes. *Decision-making* has been improved for hardware by employing 3D printing to prototype and test design alternatives, thus giving more data for making decisions. Automatic unit tests are run on HIL setups before and after software updates are merged to the master branch, providing *higher level support for automation*. However, all unit tests must be developed manually, so there is an effort required there for the developers. The compilation of code generates code documentation in Doxygen automatically. Doxygen can provide information about how functions are related, which helps information accessibility and sharing. Future work could be on enabling more automatic generation of unit tests in parallel with code development or focus more on test-driven development where the passing functionality is developed after the tests.

Velocity and agility The HYPSO project is a part of a long-term strategy for establishing capabilities for developing small satellites for scientific purposes at NTNU [23]. There is thus a need for the development strategy to have a *capacity for reuse* so that the different subsystems can be used across a variety of platforms with some changes, and reused in new satellites. Introducing the different HIL setups has increased the capabilities for *Early V&V*, which has *Reduced time* required to discover bugs. In addition, the increased employment of 3D-printing technology (also a digital technology) in prototyping and the development of Ground Support Equipment (GSE) has reduced the time for hardware development through increased *Early V&V*. Having 3D-printing technology in-house in the lab has made it easier for the team to try out new designs or physical satellite architectures and prototypes. Furthermore, there is a *Higher level of support for integration* when combining 3D-printed prototypes of hardware, mature HIL setups, and test software that can emulate physical conditions such as lost packets on the radio communication link. The GitHub workflow process introduced an *Improved consistency*, together with other standards. The shared repositories enabled students to see how others write code and test, improving consistency across the whole codebase and functioning as a resource for reuse in other platforms or future satellites.

Quality The goal of introducing HIL setups and the GitHub workflow was to *Strengthen testing* and thus *Reduce defects and errors*. However, prior to the introduction of the HIL setups, the Github flow also helped with increased testing and integration into the master branch from mid-2019. There were no measures of effectiveness prior to the introduction of DE measures, and the discussion regarding effectiveness is given in Section 9.3.2. While the issue tracking that GitHub provides was not considered explicitly when choosing GitHub for version control and code collaboration, the issue tracking and discussion functionality that it supports has enabled better *Traceability* of design choices. For example, if a bug or unwanted behavior of code during testing resurfaces, it is possible to search for keywords in GitHub, find similar bugs, and investigate if similar solutions can be used to mitigate the unwanted behavior. This *Traceability* and automatic documentation can *Reduce the time* spent bug fixing for new developers who were not a part of the project at the time of the original bug. An added benefit from incorporating the design into DE tools such as GitHub, was that it required a conscious decision and discussions regarding *architecture* and *system design* (related to both Knowledge transfer and quality), and there have been three instances of refactoring of code systematically to improve the maintainability and modularity of the codebase.

9.3.2 Effectiveness of Using Agile Digital Engineering for Software and Hardware

Tailoring of Scrum

The Scrum process was tailored to include issues related to thesis work as well as product development tasks, as shown in Figure 9.6. The stand-ups have included both the hardware and the software team, and people could join either physically or with their phone or computer. Most team members have reported that stand-ups have increased their understanding of the system and sharing of information. Some students have reported that the stand-ups increased in relevance as they were working on integrating subsystems, but not so much when developing the prototype modules. Another tailoring that was done was to agree on which issues would be performed and ensure that each student had something to work on. This was needed to accommodate thesis work. Unlike traditional Scrum processes, such as the one described by Garzaniti et al. [246], the team did not agree on the functionality for each MVP to deliver at the end of each sprint. In hindsight, a better defined MVP might have improved the results by having a shared goal for each sprint, which can contribute to team cohesiveness and commitment. This is made difficult by students' competing objectives and capstone projects being individual undertakings.

Scrum Performance

Software The first sprint using GitHub kanban was held in early 2019, and apart from the first sprint, all sprints were two weeks long. The sprints started long after the software development began, and the team had a good enough overview of the functionality. The first couple of sprints had a high number of attempted points, with a high "miss-factor" of points not done (February to June). This can be attributed to the learning process and is not uncommon for new Scrum teams.

Team members mentioned that it was challenging to figure out how to score their tasks. The Scrum leader can support this process by guiding the students, for example, by referring to previous work they have done and how long it took them to complete. An ongoing challenge has been to have enough reviewers to reduce the number of points in the "Review-in-progress" column. Since the workflow requires that someone else reviews the code, there needs to be at least one other person with similar knowledge and capabilities to review the code. This may not always be available when the students' priorities are changing to consider coursework.

In April 2019, it was decided that the software team leader would be the Scrum leader moving forward and lead the sprint meeting. Furthermore, sprint reviews should include code demonstration or more accurate documentation of how an issue was closed. The team has also discussed how to agree on a "definition-of-done". This definition has not been finalized yet, but there is agreement that it should be related to the type of issue being solved. For example, issues related to these can be draft sections or chapters, and code issues could be a bugfix, a functioning module, or a function that has resulted in a PR.

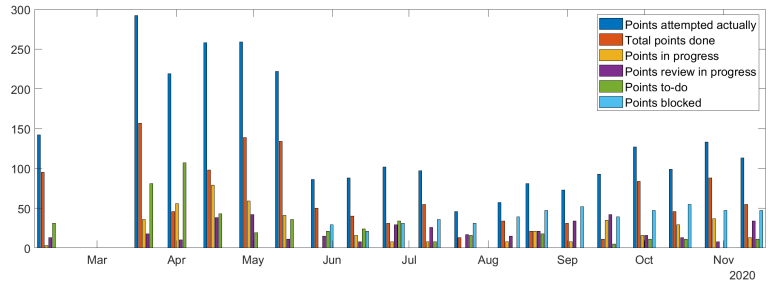


Figure 9.8.: Full SW Sprint. Illustration By Evelyn Honoré-Livermore.

Hardware The hardware team started using the agile framework and sprint methodology at the end of Q2 2020. The payload design had reached a high level of maturity by then, and most of the parts and suppliers were chosen. All satellite bus components had been procured. The remaining work was focused on verification and validation activities, and coordination with external test facilities and the in-house mechanical and optical labs. In addition, planning began for the design updates for the next CubeSat to be developed. As shown in Figure 9.9, there is a break during the summer holidays. The performance has varied over the nine two-week sprints that have been so far. Many Scrum teams take a while to learn how to estimate points to issues and how much work can be done in one sprint. Towards the end of the semester, the total points done matched the points attempted better. This could be because the team became more accustomed to the Scrum workflow, or because the deadline for delivery of the flight model was getting closer, and people felt committed to this milestone. The blocked issues were typically due to external factors, such as lack of access to testing or machining facilities, similar to the findings in [246]. There have been continuous redesign and rework activities. The stand-ups helped coordinate the activities between designers and the

group leaders organizing the support facilities. Some team members stated that using Scrum helped them prioritize tasks and not get "distracted" during the two weeks.

However, the most significant difficulties were related to attendance and commitments to sprints. It was challenging for the group leader to motivate the students when there were too few collaborative tasks. We found that a two-week duration of sprints was suitable for the team because the students were available to deliver increments in that period. Longer sprints could make it harder to motivate the students, and shorter duration would make it challenging to deliver increments [246]. The motivation could be improved by introducing stricter MVPs or by spending more time planning the work up-front. The MVPs could, for example, specify new features to be included on the hardware prototypes, iterated simulation results, increased performance, or lower manufacturing cost. The MVPs could also be tangible, for example, 3D-printed prototypes and parts that other team members can validate or simulated assembly and incremental tests.

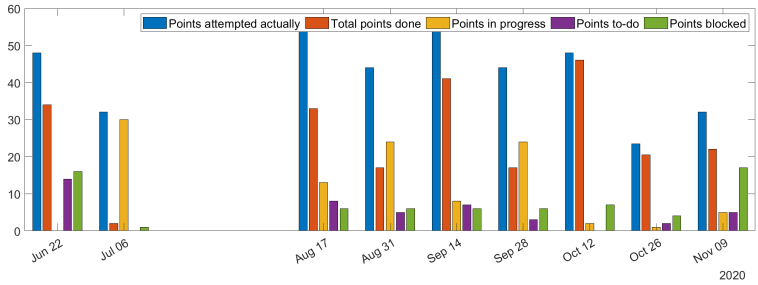


Figure 9.9.: HW sprint in barplot. There was a break during the summer holiday. Illustration By Evelyn Honoré-Livermore.

Lessons learned The team experienced challenges with commitment and attendance at stand-up meetings, especially with team members who started during the COVID-19 lockdown (fall of 2020). There were fewer on-boarding and team building activities than previous years, and little or no chance of face-to-face meetings. Some students used the Kanban board to organize their work, but did not join many of the stand-ups. Based on this experience, we see that it is not sufficient to have good workflows and tools alone, but that the social aspects also matter. The team members need to be a part of the culture, and people need to feel that they are a part of the team, which is consistent with findings of Garzaniti et al. [246] and Masood et al. [237]. The HYPSO team combined the sprint planning and review meeting to reduce time spent in meetings [237], and adjusted the sprint scoring and length to accommodate the overall school schedules and workload [237, 246].

Integration, Verification and Validation

The HIL setups have been instrumental in easing integration between different systems, both for software development, operations development, and hardware development. For software development, the HIL setups facilitate not only verification of software changes before merging with the master branch, but also verification that the changes work with the satellite bus via local

engineering model versions of subsystems or the FlatSat. There have been HYPSO-initiated interface changes, and NanoAvionics (the satellite bus provider) initiated interface changes. These interface changes have been to improve performance or add functionality. By having a HIL FlatSat-setup with physically distributed subsystems, engineers in Vilnius could update the modules remotely and work concurrently with HYPSO project members. Challenges with the HIL setups included finding people to work on setting them up and developing required functionality, such as automatic tests, and maintaining them. It was also challenging to find sufficiently exciting thesis topics for working with HIL and testing activities.

The HYPSO project team can choose which subsystems they need to locally with the payload (as shown in Figure 9.7), and which subsystems can be located at the supplier premises. The subsystems located at supplier premises can quickly receive hardware upgrades without shipping modules back and forth. Additionally, the distributed system still allows the supplier to log in to subsystems located in the university to perform software upgrades, configuration changes, or other fixes.

For operations development, the HYPSO operations' developers have been able to perform rehearsals to validate that the software functions and performs as expected. This has been enabled by allowing the operator to connect to the HIL LidSat setup using the *hypso-cli* user interface (as shown in Figure 9.5). Experiences from the operators were critical for preparing the first official software release for deployment on the flight model. This is discussed in chapter 11 as well.

COVID-19

The COVID-19 pandemic caused the university to lock down on March 12th 2020. Luckily for the team, the HIL setups had been implemented at the end of February, which allowed for remote access and testing of software on target hardware. In addition, the regular stand-up meetings had begun the year before, and only required a shift to a completely virtual meeting. The stand-ups were a bit longer than they had been previously, because more people joined regularly and there was a need to move some of the informal discussions that usually take place in the physical lab to the stand-ups. Team members also said they appreciated the stand-up meetings because they were a social interaction forum. The issue tracking on GitHub for software helped to follow-up the work, monitor the project's progress, and was not affected by the lockdown. There was an increase in commits to the main software repositories around the time of the lockdown, and the high frequency persisted until the end of semester, as shown in Figure 9.10.

However, no hardware integration and testing could be performed during the lockdown, since the team members were not allowed to travel to external test facilities. This created a severe schedule delay to the project. The hardware team spent time preparing design documentation and refining test plans until the lockdown restrictions eased.

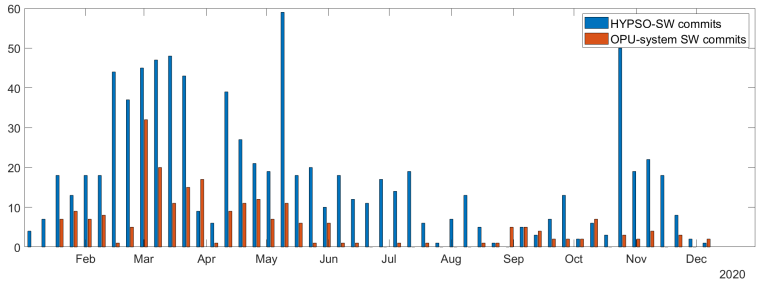


Figure 9.10.: The two main software repositories commit frequencies. Illustration By Evelyn Honoré-Livermore.

9.3.3 Educational Aspects

In the context of digital engineering, the HYPSO project organization described in this chapter has many similarities with the projects described in Berthoud et al. [22]. The university CubeSat project format is an inherently interdisciplinary project which prepares students for future work, even if it may be in different industries. Additionally, the use of HIL setups, a strict GitHub development flow, and agile practices in software and hardware development provide the students with a more extensive skill set for future employers. The students gain practical experience using digital engineering methods and tools, while still delivering the required coursework and thesis work. While these skills may be gained through capstone courses as well, having an active "customer" with strict deadlines and objectives in addition to educating students, can motivate teams to work even harder with delivering results [238]. The customer for the HYPSO project was the group of scientists who needed the data from the CubeSat, and the deadline was set by the commercial launch date. However, managing CubeSat projects with agile practices requires coordination and training, and should not be underestimated [236–238].

Although we have not done a systematic study of the transferability of skills learned during the HYPSO project, one student mentioned that:

> I have noticed that in my job, where they use Scrum with Kanban on a digital platform, I at once felt at home and prepared for how to do my work. And I also felt that I could contribute fast. The meeting structure and documentation (templates, as-built documents, internal and external design reviews) were similar to how we did it in the HYPSO project, which made it easier for me to see the value of what I had learned and realized the relevance of the HYPSO practices. (...) I felt I was prepared to start a job because I know how the workday is structured and how to organize my work.

Some of the graduated students have joined the team as PhD candidates and taken on leadership roles. The rest of the graduated students have joined companies in various industries, and some still join HYPSO design reviews or contribute to the code repository.

9.4 Conclusions

Digital engineering is needed for managing the development of complex systems. This requires a conscious effort throughout the organization, and the strategy must be tailored to the specific needs and constraints. There is also a need for engineers who are trained to use digital engineering approaches in their work, in all lifecycle phases of a project. Academic CubeSat projects provide an arena to train future engineers by collaborating in interdisciplinary system development. The students gain both technical and non-technical professional skills. For academic CubeSat projects, the needs for a digital engineering strategy are often similar to the industrial setting, but the context and constraints are quite different.

In this chapter, we have described the case of HYPSO-1, an academic CubeSat project in Norway, where we are developing a scientific 6U satellite and ground segment. Because of the challenges with knowledge sharing, unclear decision-making, lack of coordinated planning, and poor code quality and documentation, the project organization introduced measures including digital engineering tools and methods. We have outlined the project development lifecycle and highlighted how agile practices supported by a digital kanban, a GitHub workflow, and HIL setups have been essential in managing the development of the complex CubeSat. In addition, we have discussed in which ways the digital engineering strategy chosen contributed to verification and validation activities, integration of systems, knowledge sharing, and how the tools and methods supported development even during the COVID-19 lockdown. However, the tools and processes alone are not sufficient for the adoption of the DE work environment. People need to be encouraged to use them, and social aspects such as team cohesiveness and commitment are essential. Throughout this process, the project manager has used a participatory approach in which all team members could influence the practices and processes.

The digital engineering strategy adopted by the HYPSO team is a low-cost, low-effort approach using readily available tools and methods. Some methods, such as agile practices and software repositories, have been used in other CubeSat projects. There are valuable lessons to be learned between different academic teams and between industry and academia on how to best approach and implement digital engineering in the organization. Future work will look at including more MBSE tools and incorporating them with the product lifecycle proposed, to increase the common understanding of the system and support knowledge management. Lastly, it is common to simulate the hardware responses to combat the hurdles that using target hardware for testing can cause. The caveat will always be that the addition of mocking software and the addition of unit tests will be prone to the same coding mistakes as any software development. The additional overhead of producing and maintaining a mocking library can take away resources from code development that would otherwise provide the needed functionality or enhance it. The addition of unit tests should be added when possible, and could help uncover undesirable side-effects of any proposed changes to the code base.

Future studies could look at: (1) how the graduated studies have experienced transferability of skills and practices gained during the HYPSO project; (2) how other university projects use DE and how the students experience it there; (3) opportunities for cooperation between the CubeSat project and the broader university context, for example by introducing aspects with DE as a part of the student curriculum to prepare for joining cross-disciplinary projects.

ACKNOWLEDGEMENTS

The authors would like to acknowledge all the team members at the NTNU SmallSatLab for their participation in the CubeSat project. Authors also thank all the professors and technical support staff at the university departments for their contributions with theoretical insights and practical solutions to developing a CubeSat. E.H-L. would like to thank Vitech Corporation for making the CORE/GENESYS software available for this work.

Chapter 10

Software Development and Integration for the HYPSO-1 Payload

Almost everything we know about good software architecture has to do with making software easy to change

Mary Poppendieck
Implementing Lean Software Development: From Concept to Cash

This chapter presents the software architecture, development, and integration of the COTS based hyperspectral imaging payload onboard the HYPSO CubeSat. The chosen service-oriented software architecture provides a modular design that is planned to aid future development. Furthermore, this software architecture is intended to support in-flight updates of the onboard image processing capabilities of HYPSO. A perspective of the benefits of the software architecture for a general CubeSat subsystem is also given.

The strengths and weaknesses of our development procedures for software are presented and discussed. An issue tracking system to report defects, propose new functionality, and other structural changes have been used during development. The issues related to defects or bugs reported during development were analyzed and categorized to understand better what issues typically arise for this type of project. The findings from these issues indicate the importance of early testing, code reviews, and the continuous availability of target hardware for successful software integration when relying on a modular design.

The HYPSO CubeSat, with the Flight Model (FM) given in Figure 10.1 is the first scientific satellite developed by the NTNU SmallSat Lab. It is currently scheduled to be launched in January 2022. Details of the mission concept and justifications can be found in chapter 2. The throughput of hyperspectral sensors is often limited, due to the data volume, by the available communication links. Thus, an OPU with a FPGA that allows for a modular architecture is used to process the

hyperspectral image cube more efficiently in terms of data size reduction, power consumption, and operational time, when compared to just using a CPU.

Next, we briefly describe the software development and integration of the payload, which contains the OPU with an HSI and a RGB imager.

10.1 HYPSO-1 Project Organization

By the definition used in Berthoud et al. [22], HYPSO is a university space project. Student continuity is often stated as a significant challenge for university-led CubeSat projects [22, 252]. Within the HYPSO-1 project, students from different BSc. and MSc. programs provide contributions to the project mainly as part of their thesis, curricular projects, and as summer interns. The team also consists of Ph.D. candidates and researchers who manage, develop, test, and provide continuity. There is a need for project management support when developing CubeSats [22], which includes crucial documentation to transfer knowledge and extensive reviews, in addition to the development.

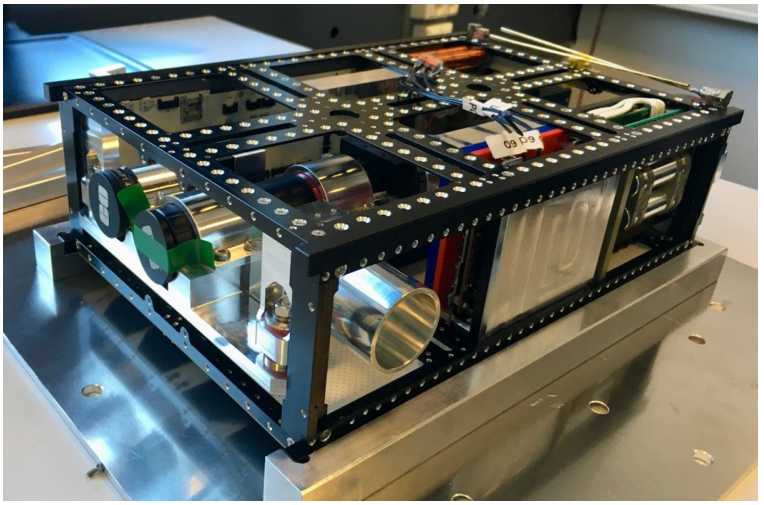


Figure 10.1.: Partially assembled CubeSat (NanoAvionics M6P platform) FM. Photo by Elizabeth F. Prentice

10.1.1 Scientific Software Development

Scientific software is used for the "analysis, design, testing, and deployment of software applications for scientific purposes [232, p. 1]". Software applications are developed in tandem with the research case study, algorithm development, or experiment, although some scientists may lack the necessary training in software engineering practices. This can hinder the effectiveness of the effort spent [232, 253]. Heaton et al. [253] reviewed scientific software development practices and highlighted that scientific software is often large, complex, and long-lived and may outlast

the researcher — leading to problems with knowledge management caused by turnover. These challenges may be mitigated using *best practices* for software development, such as documentation [254], refactoring [254–256], issue tracking, version control [255], peer code reviews, and design patterns.

Surveys from other university CubeSat projects recognize that software development and integration is often a challenging and time-consuming activity [21,22]. Reconciliation of software development with system engineering practices requires a conscious effort, and choosing and implementing the correct system lifecycle model can be challenging [257].

10.1.2 CubeSat Software Architectures

Software architectures can generally be divided into: *state-machine* types, *centralized architectures*, and *distributed architectures* using messaging systems, as described in the CubeSat flight software architecture review in [258].

State-machine-type solutions offer a simplified implementation of the flight software when the functional requirements are well defined. Still, a change in requirements or a discovery during development can affect the states and transitions in unforeseen ways, and it can be challenging to retain modularity [258]. Centralized architectures are used when there are hard real-time requirements [259, 260]. The HYPSO CubeSat is constrained by the Application Programing Interface (API) available for the COTS camera, which requires an embedded Linux OS [260]. This OS is not designed to have real-time processing capability, and does not guarantee that specific processes are completed within a given deadline or at a specific time. Distributed architectures such as a Service-Oriented Architecture (SOA) are more flexible solutions to support an incremental development or changes in requirements, when the services are independent [258]. For SOAs, the *request-response* pattern is common when developing Cubesat Space Protocol (CSP) applications [260], and adds more flexibility and demands less coupling between modules compared with centralized architectures [258, 260]. The requirements for the HYPSO software system are detailed in Table 10.1. The system needs to be modular and extensible to be able to utilize the available contributors fully, and this led to the decision to use a SOA.

10.1.3 Contribution

The remainder of this chapter presents the HYPSO mission perspective, which determined how the software architecture was designed and developed to accommodate the requirements described in Table 10.1. The corresponding management model was adapted to support the development and integration-related challenges. We tailored digital engineering practices to suit the university context [261]. In addition, we chose an architecture to enable the development of the software as modules of services and features and focused on early integration of these. This software development approach demonstrates one way of developing a CubeSat Payload with similar resources and challenges as found in the HYPSO mission.

Table 10.1.: Requirements for HYPSO-1 CubeSat software system based on [258].

Feature	Need
Extensibility	The HYPSO software system shall allow for new functionality through in-orbit upgrades.
Modularity	The HYPSO software system shall allow separation of functionality to enable concurrent development amongst the team members.
Reusability	The HYPSO software system and modules shall be reused on different assets (such as UAVs and multiple spacecraft).
Testability	The HYPSO software system shall enable compilation for different CPU architectures
Reliability I	The HYPSO flight model installation shall have a "golden image" with basic functionality that is redundantly distributed on the payload hardware, for recovery from software failures.
Reliability II	The system shall have watchdogs supported by the spacecraft bus in case of Single Event Effects (SEEs) or other temporary malfunctions.
Reliability III	The *boot loader* shall be protected against accidental modification.

10.2 Software Development Process for Hypso-1

The mission and software development was distributed between the satellite bus manufacturer and the NTNU team. Because of this, the satellite bus manufacturer provided a remote FlatSat [22]. A FlatSat is used to mimic the hardware and software of the actual CubeSat. This remote FlatSat was used to integrate with on their premises, such that the integration and testing of the payload with the other subsystems could continue during the lockdown imposed by the covid-19 pandemic [261]. Partly due to the pandemic, remote software development and testing were facilitated for target hardware of the payload while it was connected with other subsystems of the 6U satellite bus through the remotely connected FlatSat. Due to the already planned infrastructure for remote integration, this eased the consequences of the pandemic for the software development when compared to the hardware development, which was more affected [261].

Specifically, multiple replicas of the OPU were connected to a FlatSat with the same electrical interfaces to be used in the Final FM of the CubeSat. This enabled development and rapid testing on target hardware in an environment that provided continuous integration. Testing is repeatedly emphasized as a high priority activity when developing CubeSats [22], as it helps reduce the time from initial code development to deployment and testing [21]. A more detailed description and discussion of the testing setup is given in 11.

The details on how digital engineering tools were used during software development are given in [261]. In short, the adapted agile practices have relied on a tailored Scrum approach. This was enabled using digital engineering tools provided by GitHub™. Sharing experiences and knowledge management, coupled with a well-defined and straightforward workflow, enabled improved verification, validation, and integration activities. However, some of the challenges typical

for university-led satellite projects, specifically resource management with multiple competing objectives, were still prevalent but less so. Using the digital engineering tools, the different activities, both development, and academic work, were given a common source, helping to compare and prioritize them.

10.2.1 Software Lifecycle

The HYPSO mission is intended to span several years, during which multiple satellites will be deployed in a constellation. As a result, the software development should continue over that period to enable bug-fixing and added functionality through in-flight updates. The choice of software architecture needs to enable both. The benefits of developing reusable software are significant. The basic software lifecycle from ISO 24748 [262] and the different phases for our project are shown in Figure 10.2.

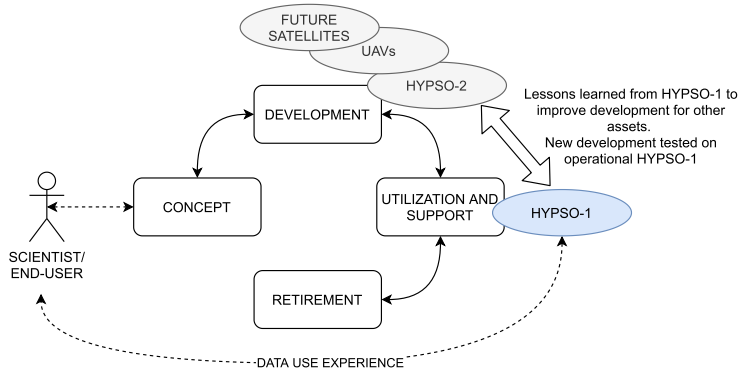


Figure 10.2.: Software lifecycle based on ISO 24748 with intended use shown [262]. Illustrations by Evelyn Honoré-Livermore

Software Concept Phase The operators and scientists provide the functionality requirements of the software system. For example, scientists expressed needs for specific validated data products that were allocated to functions such as camera parameter configurability and choice of specific On-board Image Processing (OBIP) functionality `obip-services` [23]. The satellite operators were not involved early in the concept phase, but joined the project after the software development had begun. Their needs resulted in interface and functional changes, which were tested and refined as a part of the workflow described in [261]. The `obip-services` are intended to be updated during the mission.

Software Development Phase Capstone projects were derived from the requirements found in the concept phase to support assignments to the development team [261]. These capstone projects were further divided into tasks for the Scrum sprints used to guide development. Development

included implementing new features as their need became apparent, bug-fixing as errors were discovered, and refactoring when appropriate.

Through bi-weekly sprints [261], the developers needed to resolve that each identified issue or task was estimated, and goals for the following sprint were set. At the end of each sprint, the progress was reviewed and a new sprint was planned for the next period.

Software Utilization and Support Phase Not all incremental software improvements result in a set of services that will be used in-flight. Nevertheless, there is an underlying goal of doing rudimentary testing of each proposed code contribution so that the most recent version of the payload software is working. More extensive testing with all available subsystems is performed before making a flight-viable release. In this process, we are utilizing semantic versioning [263], namely a standardized method for documenting changes between versions. This provides a high-level description of the changes that can be more easily communicated to other team members and software users. Semantic versioning also provides guidelines on how to ensure that changes in API are avoided or accounted for and communicated correctly.

Software Retirement Phase It is foreseen that some modules will be re-used in other systems, and the knowledge management in GitHub™ can support this. With the high personnel turnover often found in CubeSat projects, it is important to have good knowledge-transfer to ensure progress [22], such that modules can be re-used and the same functionality is not developed multiple times. To facilitate this, the future systems are planned to utilize the same or similar COTS components while only making incremental improvements.

10.2.2 Payload Software Architecture

As defined earlier, HYPSO employs a SOA. This allows for services to be provided by application components to other components (both within the same subsystem, and with other subsystems) via network communication. A *service* is here defined as a set of related functionality that can be requested from the user [264].

To interface with the chosen COTS camera, the supplier's official closed source API is used, which constrains the selection of an OS to be an Embedded Linux system. However, this OS is a well-established and widely used OS, and does not come with the unknown bugs of a custom build OS, which can be common for CubeSats [265]. The OS image of the payload is therefore built using the open-source PetaLinux build system from Xilinx to deploy an Embedded Linux system [259]. The chosen Zynq 7030 System-on-Chip (SoC) with the FPGA used for the on-board data handling is supported by PetaLinux [260]. The build system was customized to include the payload application, and necessary drivers, as applications within the OS image [259].

The software is here defined as executable applications started on boot. These applications are the service providers of the payload, as shown in the simplified diagram of Figure 10.3, and they

are planned to be updated in-flight. The File Transfer (FT) service is made to interface with the satellite provider's API and Interface Control Document (ICD), and is compatible with both the payload and the rest of the satellite bus subsystems. The HSI and RGB services send commands to the respective cameras. The CSP and OS communicates with the other processes and provide Linux commands, respectively. The Telemetry service logs the state parameters of the payload over time.

By separating the functionality as a stand-alone executable, we can restart the subsystem into a known state at each power on. There is no persistence in the OS image so that it will experience a fresh start at each boot. In addition, the executable service provider can be hot-swapped during execution. This means updating the software by simply uploading a new file without altering the booting sequence. This enables us to have multiple service provider versions available during operations. This addresses the first requirement in Table 10.1 about extensibility, an uncommon feature of traditional spacecraft systems.

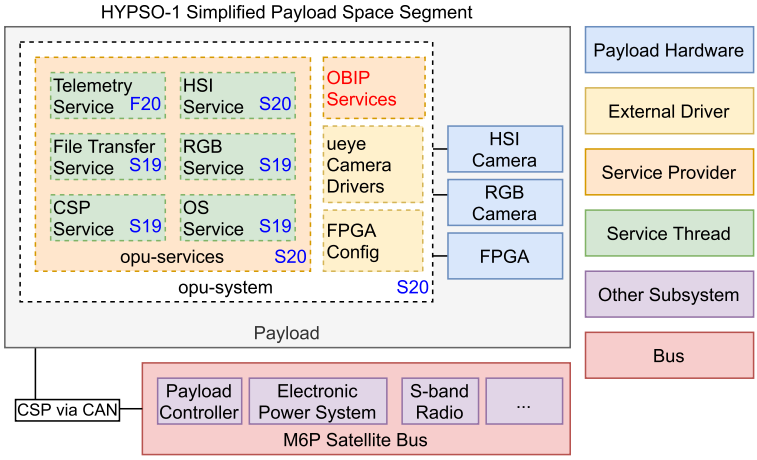


Figure 10.3.: Simplified overview of the HYPSO payload with dashed boxes as software components and solid boxes as hardware components. The season and year for integration of the module are given as blue text.

Firmware is defined as the system image, or `opu-system`, which consists of the kernel, device tree, root file system, and the FPGA programmable logic bitstream [259]. An updatable primary image is stored on an SD-card, and a backup is stored in embedded memory. This backup shall not be changed after FM assembly. The difference between the firmware and software update process is primarily due to the amount of data that needs to be transmitted for an update, and the level of risk associated with performing their updates. Backward error recovery for firmware updates is made by restoring the image from embedded memory, if the primary image fails to load after attempting to boot a set number of times.

The root file system is always loaded from the boot image. This ensures that the system enters a known state after boot. Payload data and different versions of software applications are stored on an SD-card, and can be taken into use at run-time. The planned updates will add new versions of the service providers [23] with new algorithms and bug-fixes.

The `opu-services` application/executable is the minimum viable product. This application captures, compresses [78] and transfers hyperspectral and RGB images, communicates with other subsystems, provides telemetry information, and provides remote shell access to the OPU. A version of `opu-system` and `opu-services` that passed all our tests were integrated into the payload FM and shipped to the satellite bus provider for integration with the rest of the satellite bus during the early summer of 2021. Software intended to expand the capabilities of the payload will be added during the mission lifespan.

A planned future application is the OBIP, which will process and derive high-level low-latency data products from the captured data [23, 266], such as results from classification and target detection. The application is marked in red to indicate that it is a part of a future extension of the `opu-system` in Figure 10.3. With this proposed SOA, we expect to be able to update only the software containing the services available from the payload with low risk to the mission, when compared to updating the entire system image. With this modular and extensible design, code-contributors can add to the system in the form of a single application or by expanding the services available from an existing application on the platform. Updating the `opu-services` application entails risks that are necessary to meet changing processing requirements.

The application repository (*hypso-sw*), in addition to deploying the payload executable known as `opu-services`, also builds the `hypso-cli`. `hypso-cli` is the Command Line Interface (CLI) used to send commands via CSP packets that will be propagated from the mission operator to the satellite bus, and subsequently to the payload subsystem [260], as indicated by Figure 10.3. CSP is a lightweight network protocol, resembling IP, that can be used as a network layer between both physical subsystems and between different software services internal to one subsystem. CSP supports several hardware layers, where HYPSO relies on CAN and Universal Synchronous and Asynchronous Receiver-Transmitter (USART). The `opu-system` starts the `opu-services` in the version packaged into the image on boot [259]. The system supports having multiple versions of `opu-services` in non-volatile memory (SD-card) and can change between them at run-time. New versions, with bug fixes, can then be added without compromising old ones.

10.3 Software Issue Analysis

We aim to understand better what kind of problems occurred in the development and integration process of the satellite and how we solved a subset of them. To do so, we encouraged the team to document discovered bugs, desired features, and other proposed changes as GitHub™ "issues". Then we surveyed all open and closed issues labeled as *bug* that resided in the repositories and separated these into four categories based on how they were discovered:

- When interfacing between subsystems (*Sub.*),
- When testing the internal functions of a module or interfacing between services for the payload (*Mod.*),
- When considering scientific data handling (*Sci.*),
- or miscellaneous (*Misc.*).

The categorization was done in two rounds; first independently by three of the authors, and then as a group with a clarification of interpretations of the categories. The bugs related to FT, use of external drivers with their underlying errors, and general subsystem communication from the perspective of the payload were classified as *Sub.* bugs. Issues related to the functional behavior of modules and their internal communication were classified as *Mod.* bugs. Issues related to data handling were classified as *Sci.* bugs. The issues that did not fit any of these categories, e.g., virtual build environment, were classified as *Misc.* bugs.

Table 10.2.: Categorized issues labeled as bugs.

Repository	# of issues	% Closed	Sub.	Mod.	Sci.	Misc.
hypso-sw	78	76.92	27	39	2	10
opu-system	22	100.00%	2	15	0	5

The findings from over two years are given in Table 10.2 and show that most of the issues reside in the `hypso-sw` repository. This is also the repository that has had the most contributors. Here, the issues are found mainly in *Sub.* and *Mod.* bugs, with the other categories being less prominent. For the *Mod.* bugs detected the FT service is a frequent source of discovered bugs. This FT service was developed at an early stage of the project, as seen in Figure 10.3, and has been invoked frequently when testing the functionality of other services as well.

The HSI functionality was developed as a standalone subroutine before being integrated as a service within `opu-services`. If the focus had been on earlier integration of the HSI service, there would probably have been fewer challenges to resolve during integration. The CSP and OS services rely on third-party implementations with larger communities, and fewer or no bugs are related to these. The RGB service is smaller/simpler and has been invoked less during testing, and has fewer bugs related to it. The telemetry service is a late addition to `opu-services`, and other services do not invoke it, and few bugs are related to this service.

Automated tests helped discover many issues. However, most of the issues reported were not directly related to the code under review. That is, the exploratory nature of manual testing made it possible to discover issues beyond the scope of what was initially supposed to be tested.

Other issues in `hypso-sw` are related to feature requests and other enhancements.

10.4 Refactoring and Future Missions

In software development, refactoring is defined as "a change made to the internal structure of the software to make it easier to understand and cheaper to modify without changing its observable behavior." [256, p. 565] This enhancement activity has been performed to make the code base more maintainable and accessible for future contributors. As an example, the HSI service has been refactored multiple times to better divide its functional parts into smaller, more maintainable functions. This has made it easier to understand without changing its behavior. By this refactoring, the performance of some routines was improved, thereby providing performance gains for the arguably most important service of the payload and thus delivering a more capable mission.

A second satellite is planned; namely, HYPSO-2 [267]. This satellite is intended to feature an additional secondary payload, a Software Defined Radio (SDR). While this payload also is based upon a similar hardware platform as theOPU, it will have its separate system image and application image. Parts of the codebase have been refactored to support better the `hypso-sw` for multiple payloads. This relates to the application services that integrate the payload with the bus, such as shell, CSP and FT services in addition to the telemetry services. The SOA and a common OS running on both payloads made this possible. Successfully supporting this development within the existing *hypso-sw* repository is a demonstration of the extensibility, modularity, and reusability of the software as specified in Table 10.1.

The chosen SOA provides a high-level abstraction that enables further development of the service modules of `hypso-sw` and development of new services for future satellites. The first satellites will also benefit from future development as they can be updated. Modularization, refactoring, and generalization have been the fundamental principles used to meet the driving needs that are defined in Table 10.1 [267]. Several common factors for future payloads were identified through the refactoring process to ease future development further.

10.5 Discussion and Conclusion

The software development process for the HYPSO project provided insight into how such systems could better be developed and integrated in the future.

Relevant literature emphasizes the importance of testing [22]. Experiences from developing the HYPSO software made it clear that testing and integration will help with discovering errors, as more testing of a given service uncovers more bugs. It can be challenging to find the human resources to test extensively, and this activity can be challenging to motivate in a university setting [22, 261]. Integration with other modules or subsystems is also prone to introduce new insight. The analysis of issues registered for the HYPSO software given in Section 10.3, with a focus on those labeled as bugs, also substantiates frontloading of testing, and early integration, as recommended by [22].

The choice of software architecture made it possible to develop functionality as independent components or services. This helped generate several contributions from multiple contributors with high turnover, without adversely affecting the functionality or development cycle.

The separation of platform and application, Embedded Linux OS and `opu-services` respectively, was done to make it possible to perform in-orbit upgrades with lowered risk to the mission. Upgrades of the system and software have yet to be demonstrated in-orbit. However, this upgrade functionality has been tested extensively, e.g., as a part of software development. That is, new software contributions were regularly developed on the target hardware, and the upgrade functionality was used to deploy and test them.

The development and integration of future services for new payloads have demonstrated the perceived benefits of the chosen software architecture [267]. The SOA enables the reuse of code for future development.

These experiences will aid in the development of future satellites that are planned by the NTNU SmallSat Lab and can be scaled to other similar systems as well.

Acknowledgment

The authors would like to thank all the MSc. students contributing to *opu-system* and *hypso-sw*, especially M. Danielsen, M. Hov, T. O. Moxnes, and J. A. Gjersund. We thank the numerous reviewers and are grateful for the participation of NanoAvionics and their continuous feedback throughout the project.

Chapter 11

Testing of the HYPSO-1 Payload

> *If you don't have questions about a product's risks, then there's no reason to test. If you have at least one such question, then ask: Will these tests cost more to execute than their answers will be worth?*
>
> Gerald M. Weinberg
> *Perfect Software: And Other Illusions About Testing*

The growing community of CubeSats vendors makes it possible to launch and fly novel payloads for targeted applications by procuring flight-proven CubeSat platforms. The importance of embedded software for such COTS based payload systems has increased to provide more functionality and flexibility. As COTS components are not designed for space, they warrant extensive software and hardware testing. The ongoing work on how the payload software testing procedure is used in the development of the HYPSO satellite is given in this chapter. This chapter discusses the software development strategy, the challenges that were encountered, and the lessons learned throughout the process. In particular, the advantages of rehearsals, reviews, and manual testing are compared to different methods for automated and programmer-driven testing.

The subsequent sections present the experiences and findings from software system integration testing of a HSI payload for a university CubeSat mission. The HYPSO spacecraft is primarily a science-oriented technology demonstrator enabling low-cost and high-performance hyperspectral remote sensing and autonomous onboard processing to collect ocean color data products in collaboration with other autonomous platforms [23, 268], with details in chapter 2. The planned image processing pipelines are still under development at launch and need more testing.

Contribution

Here we present how the HYPSO team performed the testing of the payload software, and how testing strategies were adapted to support our development and integration challenges. These adaptions made the payload software testing feasible, and possibly adequate, for the size of the

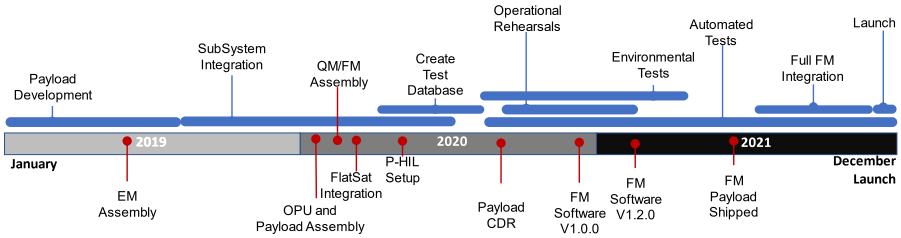


Figure 11.1.: Relevant development and testing timeline for HYPSO

team and the mission's acceptable risk. The testing focused on the early use of target hardware and enabled early integration [22]. Rehearsals, reviews, and manual testing are compared to automated and programmer-driven testing for our system.

Section 11.1 presents the background and related work. The testing strategies are described in section 11.2. The results from the testing process are described in section 11.3. A discussion of the findings and lessons learned are given in section 11.4. Lastly, in section 11.5, we provide conclusions.

11.1 Background and Related Work

The CubeSat standard is a simple small satellite reference model [269]. This standard is intended for low-cost satellites with a short project cycle relative to traditional space missions, both in terms of development, launch, and operations [20]. A CubeSat is based on the form factor of "cubes" or "units" with each edge measuring 10 centimeters. These units can be combined to create satellites with different form factors, e.g., 1U, 3U, 6U, or larger [20]. With this mechanical envelope standardization, it is possible to launch several CubeSats from a single launch vehicle, and there are multiple CubeSat spacecraft and subsystem vendors available. Due to the standardization of the mechanical interfaces, a CubeSat team can freely base their choice of vendor.

In this chapter, the focus is on the development and testing of the HSI payload as an onboard processing platform, as well as its integration with the rest of the satellite bus provided by the CubeSat vendor. The *payload processor* or OPU receives and transmits commands and telemetry, as well as raw or processed payload data, to other satellite bus subsystems. The OPU does not play a direct role in spacecraft telemetry and telecommand data management. The CubeSat vendor provides those subsystems. This simplifies the design of the CubeSat and provides more resources in the development of the payload [20].

The European Cooperation for Space Standardization (ECSS) body of standards encourages reviews to assure better documentation, and states it as an opportunity for stakeholders to get an overview of the system details [269, 270]. Quality is assured in space missions by *reviews* and *tests* [22, 269]. Tests should demonstrate that the implementation meets the functional and performance

requirements. CubeSat projects often discover unexpected and undesirable behavior during testing and integration, and the importance of these activities are strongly emphasized [22, 271].

In a university setting, there are challenges related to team organization, such as availability of (qualified) students at the right time as well as a high turnover [22, 271, 272]. This influences the whole development process, from design, implementation, test, and delivery. For many university teams, it isn't easy to successfully transfer knowledge to new team members. Through trial-and-error, the HYPSO team adopted a version of an agile digital engineering workflow described in [272]. The software architecture design to accommodate these challenges is not covered here but in chapter 11.

All team members are encouraged to contribute to testing to mitigate the lack of a dedicated test team. This makes adapting and implementing straightforward and efficient development and test routines even more critical.

Complex On-Board Data Handling systems are typical for technology demonstrating CubeSats, where more services are performed onboard. This requires more software to be developed to interface, control, and operate different subsystems, which leads to a need for more testing activities.

In the literature, several approaches for software testing and integration activities of CubeSats have been reported. The focus is often on Software In the Loop (SIL) [22, 269, 273], Model In the Loop (MIL) [269, 271], and/or HIL [22, 269, 271, 273]. SIL focuses on procedures and functions for isolated units and checks for the expected input/output relationships. MIL focuses on simulation models and identification of expected communication flow and interference. HIL focuses on the behavior of the software executing on the target hardware, the use of target hardware-specific functionality, and communication with other subsystems. It is strongly encouraged to use any means possible to enable testing, preferably on target hardware with a focus on early integration [22].

Through this chapter, we focus on the testing activities used by the HYPSO team to ensure the desired functionality of the software deployed on the OPU and finally integrated into the CubeSat. A summary of the testing strategies and their focus is given in Table 11.1.

Table 11.1.: Testing Strategies, with details in Section 11.2.2

Focus	Category
Automated workflows focusing on CI for development and deployment	HIL SIL
High availability of target hardware for development and subsequent testing.	HIL
The development of test suites simulating nominal operations	HIL MIL
Rehearsal with trained and untrained operators	HIL MIL SIL
Post-launch Testing and Development	HIL MIL SIL

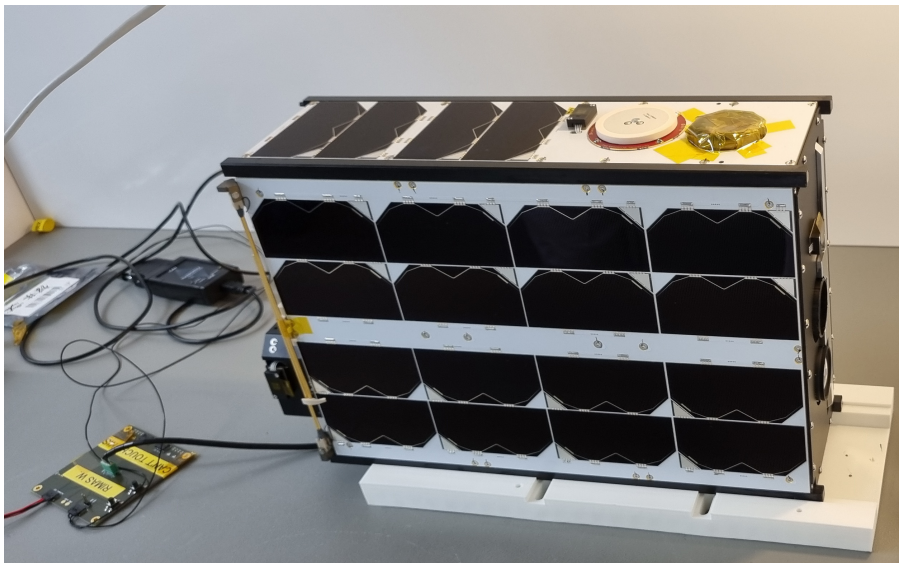


Figure 11.2.: An image of the 6U CubeSat going to space during fi al system check at the CubeSat vendors facilities.

11.2 Embedded Software Testing

Ideally, the payload subsystem is tested in an environment as close to flight as possible. Here, parts of the operation are simulated, e.g., procedures for operation and flight, and only the satellite's subsystems directly relevant for the payload are used. Thus, it is not strictly a HIL-system, as we do not have any simulated input from all sensors and outputs to virtual actuators, but rather a Payload-HIL (P-HIL).

Embedded systems are typically tested in different ways throughout various development stages. For the HYPSO mission, we tested the functionality and performance of communication and image processing independently. The communication is tested by interfacing the payload with the other physical satellite bus subsystems. For the onboard image processing, the development algorithms usually start as a proof-of-concept prototype in Matlab or Python. If the concept is promising, it is transformed into a C program and tested on a desktop computer. This code is then ported and adapted to the target hardware, where the processor, power, and memory resources are limited, thus constraining the operation and performance. In this stage, target architecture-specific instructions have been used to improve performance when possible. If appropriate, the algorithm may be implemented in a hardware-accelerated form, employing a FPGA. This process is illustrated in Figure 11.3.

As shown in Figure 11.1, system and integration tests started when the first HIL-setup with the OPU was in place in February 2020. The testing continues as discovered bug fixes and new features are intended to be updated during flight [23].

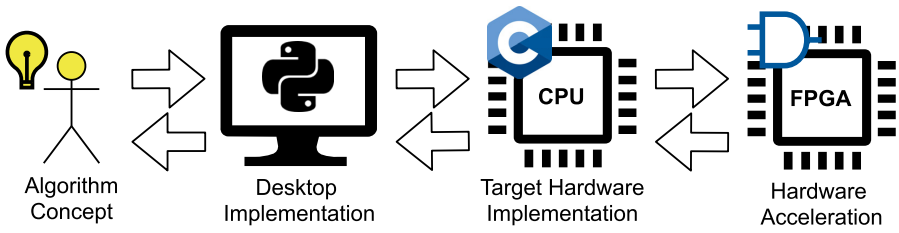


Figure 11.3.: Illustration of the algorithm development flow. If one stage passes it may move on to the next, or be revisited.

11.2.1 Testing Infrastructure

Figure 11.4 shows the basic system architecture for the fully deployed system. The Payload Controller (PC) is responsible for propagating commands and responses to and from the payload. The lightweight and broker-less messaging library used in parts of the fully deployed system is Nanomsg Next Generation (NNG). The other abbreviations are explained below, and to some extent in the appendix A.

Figure 11.5, 11.6, and 11.7 illustrates how multiple subsystems of the procured CubeSat platform are accessed. Each colored box has a unique CSP address for the given test setup. Different communication pathways can be tested by using different CSP addresses through a CAN and internet-based connection to a FlatSat at the satellite platform vendor's premises. The FlatSat is a mimic of the actual CubeSat where components are installed and connected so that it can be interfaced with remotely for testing [271].

The software propagating commands or requests from an operator to the payload uses a service-oriented architecture based on a request-response pattern. That is, the operator sends a request via the Command Line Interface application `hypso-cli` application on the ground, and the `opu-services` application running on the OPU sends a response. The satellite bus and its ground system rely upon the use of CSP, and the OPU also implements CSP as its primary communication protocol.

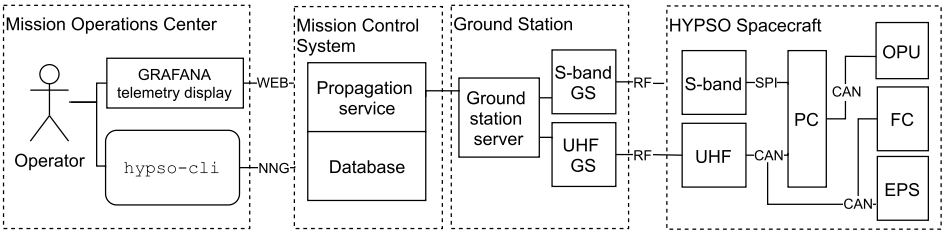


Figure 11.4.: Fully deployed system architecture. On-board Processing Unit (OPU), Flight Computer (FC), Electrical Power Unit (EPS), Payload Controller (PC), Controller Area Network (CAN), Ground Station (GS), Radio Frequency (RF), Nanomsg Next Generation (NNG)

LidSat for development

The LidSat-setup is the most complete and versatile setup. It includes the OPU and integrates with the development models and prototypes of the rest of the satellite bus. It also enables testing of communication links by UHF-radio in the test-loop, as well as the Mission Control System (MCS) environment to emulate better the expected communication interfaces between the operator and CubeSat, as seen in Figure 11.5. This is also used for the system functional test campaign here called rehearsals (also known as Test-as-you-fly [22, 271]), covered in Section 11.2.2.

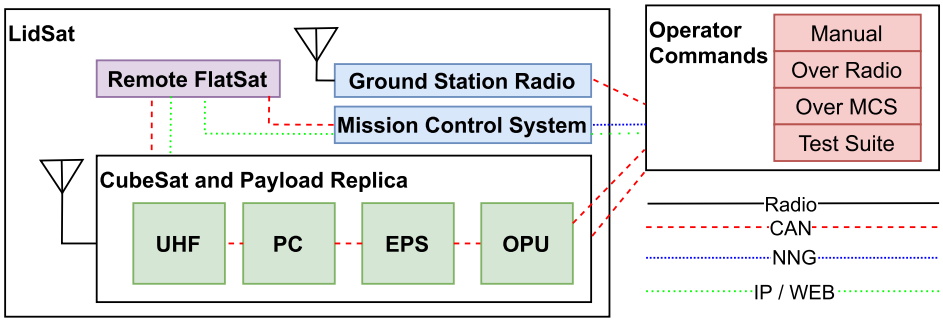


Figure 11.5.: The LidSat testing setups where different paths for command propagation is used by utilizing unique CSP addresses to test communication at different levels.

P-HIL for automated tests

The Payload-HIL in Figure 11.6, featuring only the OPU and an EPS, is mainly used for automated testing. The P-HIL setup connects to a testing automation server using Jenkins, capable of running regression tests on PRs and after branches were merged, and periodic performance tests, as defined by the requirements. It is also possible to remotely run the Jenkins tests on the LidSat, enabling automatic tests of system functionality [271].

Aid for environmental tests

Environmental test setups, i.e. the Engineering Model (EM) and Qualification Model (QM) of the payload [271] were used when the payload was tested in the expected thermal and vacuum conditions. A setup like this is given in Figure 11.7. To support the environmental test campaigns, semi-automated test scripts allowing controlled repetitions of tests were developed, based on the same tests used on the P-HIL test setup. In addition, an electrical Ground Support Equipment set, consisting of Power Supply Units (PSUs), cable harness for power, and communication was developed, as it was not deemed expedient to test the flight-proven EPS. The PSU in Figure 11.7 can be regarded as a simplified and manual EPS.

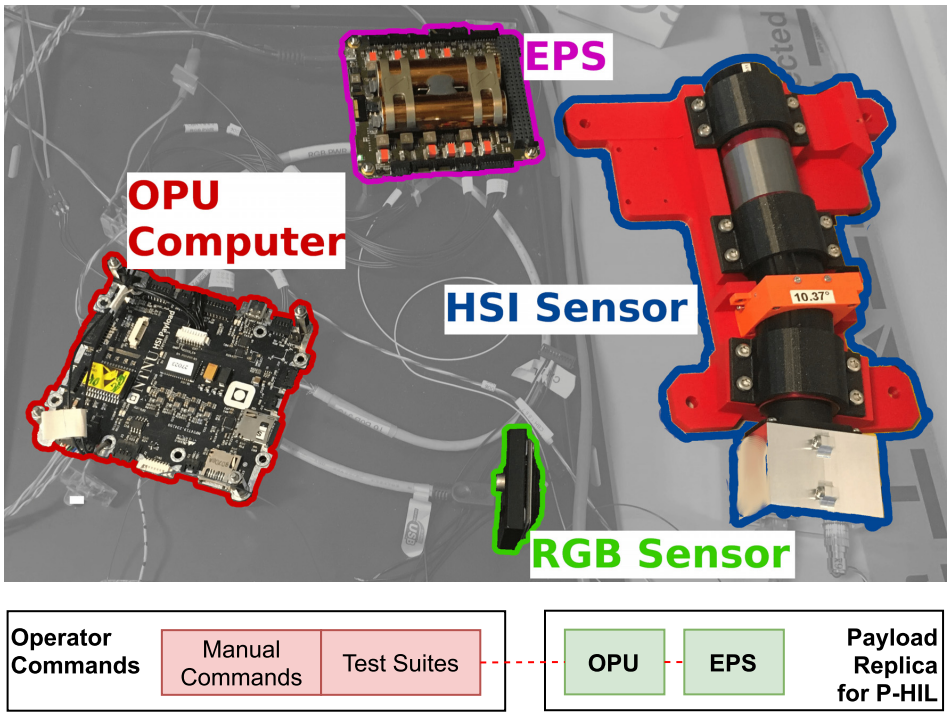


Figure 11.6.: The P-HIL testing setups where the payload and EPS is accessed via CSP over CAN marked with dashed red line. Original Photo by Elizabeth F. Prentice

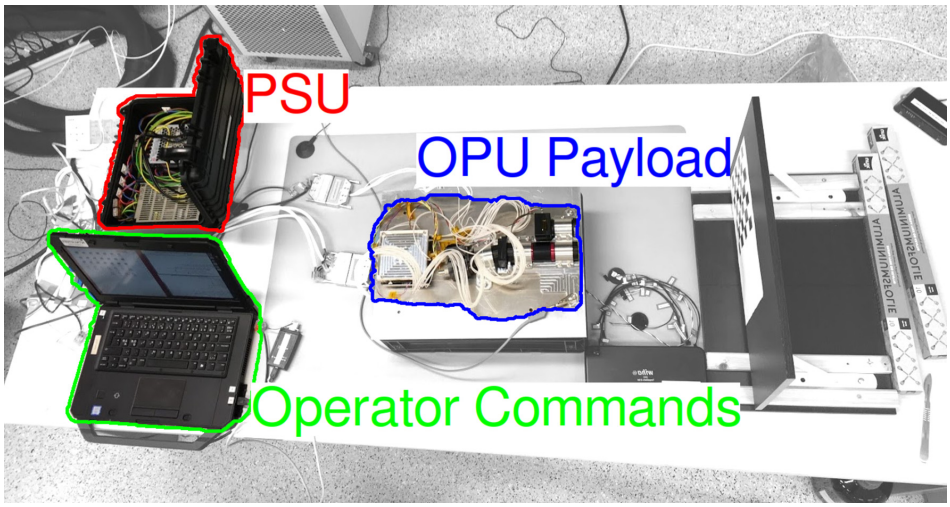


Figure 11.7.: The testing setups used for environmental validation of the payload where it is accessed via CSP over CAN, marked with dashed red line. The PSU is manually operated during environmental testing. Original Photo by Elizabeth F. Prentice

11.2.2 Testing Strategies

Several testing strategies were attempted to understand better what worked best for our team. Here are four high-level descriptions of the testing strategies that were eventually chosen.

Continuous Integration Workflows

As part of the workflow, procedures and functions are tested using the *Check Unit Testing Framework for C* [274]. That is, at each PR a set of unit tests are executed to check if the input/output relationship for the functions is still as expected. Ideally, the tests would be developed first, followed by a function that meets the defined requirements. Using this development strategy for our team has proven difficult, possibly due to the limited timeframe student developers have, and inexperience in the methodology. As a result, relatively few unit tests have been developed. Regardless, the tests that exist have helped to discover undesirable artifacts.

Target Hardware for Development and Test

Ensuring target hardware availability for both development and testing has been a priority. Initial testing is done by the developer on the target hardware, which helps to identify basic errors. When the developer regards the code contribution as ready to be merged into the central repository, they open a PR. A review of the written code and a test of the resulting executable(s) are then performed by other team members. This will prompt a discussion about whether or not the code contribution performs as intended and is maintainable. Changes can be requested before approval. The process, further described in [272], has helped us to identify errors and bugs hidden from the initial developer, as well as to clarify misunderstandings in functional requirements. An added benefit is that an overview and understanding of the codebase is distributed among more team members.

This setup made early integration tests possible, which is recommended [22]. These tests done during development, for both new functionality and regression tests, are usually manually executed with the payload connected to other parts of the satellite, either locally or through a remote FlatSat. Infrastructure functionality, e.g., subsystems communication, execution of remote commands, etc., are tested more frequently as a result. These tests focus on high-level functionality and have uncovered ICD issues, functional problems with UI, illogical programming, and shortcomings of documentation, among others.

Test Suites Simulating Nominal Operations

Automated tests, where nominal operations are simulated on the P-HIL, were also made. A *Jenkins*-server builds a new version of the software for every merged PR and tests its performance on target hardware. These test suites can also be invoked on demand. This has helped us uncover changes in performance and changes in ICDs, as well as making consistent testing of nominal operations more

available across different communication paths. The P-HIL is configured to be tested with both the LidSat and a remote FlatSat provided by the CubeSat vendor. The nominal tests were also used to test the payload during mechanical, thermal, and vacuum testing, ensuring consistent testing procedures.

Rehearsals with Operators

An operator simulates how the satellite will be interfaced with during the mission in the rehearsals. This includes enforcing constraints such as communication windows, injecting errors on communication links, and letting the operator carry out complicated procedures. Participants consisted of both people familiar with the UI and people who had to rely on documentation. These rehearsals have revealed functional issues and high-level problems, as well as uncovering procedural inaccuracies and illogical UI. It has proven multiple times to be a valuable way to test if the satellite and its interfaces behave as desired, if the predicted constraints will hinder operations, and if the documentation is sufficient.

Post-launch Testing and Development

The HYPSO satellite is intended to be updated during its lifetime. There are benefits of having a replica of the system on the ground, e.g., to test telecommands and updates in a non-operational environment [22]. We plan to use our existing test setups and strategies to test new software modules and telecommands, train operators, and verify updates. It is planned to further extend the existing testing infrastructure with more replicas of the CubeSat and Payload.

11.3 Testing Results

In our team, it is the creator of a unit of software that is responsible for creating unit tests. This can lead to rapid deployment of tests. However, a risk of this approach is to create tests that pass, rather than tests that test the current software for unexpected results.

A majority of the software testing can be automated with automated test setups. The automated testing has helped uncover changes in the CLI, as well as unintended changes in the request-response pattern between `hypso-cli` and `opu-services`. Three different test suites are run and cover nominal operations of the satellite at every code change. These test suites will then give an error if there is an interface change in the commands used for nominal operations. However, there are still some human interactions that cannot be automated. This does not reduce the importance of an automated test setup as it supports rapid deployment and regression testing. The automated test setups, i.e. LidSat and P-HIL, do not necessarily test for new and unexpected states of the system. This can lead to false confidence in the system, as a system can still fail in operations despite passing all automated tests. This potential false confidence supports the use of manual testing, as done during the code reviews. The manual testing should be extended to incorporate new tests of new functionality, but this is not done currently.

The issues discovered during code reviews were mainly concerned with whether or not the proposed changes were performed as desired. When doing these reviews, functionality unrelated to the original PR was also tested, and some issues could be reported as a result. Problems with illogical UI and limited documentation were also addressed as part of the review. The code reviews became an excellent platform to discuss the proposed changes and often led to incremental improvements from the original PR. Initially, these code reviews were highly unorganized, but a more coherent and effective process emerged when establishing a bi-weekly three-hour session for code reviews employing *mob programming*.

The rehearsals performed in the fall of 2020 uncovered several new issues and areas of improvement for the software. These issues were both quality-of-life changes for the developers and operators, but some were crucial to the mission. The rehearsals invoke more of the complete mission infrastructure and use more available communication paths than our other tests. This has been helpful to discover changes in communication performance, scheduling, planning, and timing issues and learn how the operator needs to handle errors during a satellite pass. In addition, since the communication windows have been constrained, the rehearsals have also let operators experience the stress related to performing a set of tasks within each time window. This experience also leads to the re-design of procedures and the creation of more high-level functions that aid the operator in performing tasks more efficiently. The issues resulted in, among others, the following changes:

- Improved redundancy in the power control of the payload.
- More appropriate default timeouts for file transfer between subsystems and ground.
- Improved state acknowledgment recovery during transmission loss.

11.4 Discussion

One challenge for the existing setup is the limited test coverage. Not all software units have tests implemented for them, both with or without target hardware. At the time of writing, we have 8 test suites testing different services with 46 (successful) unit tests. It would be beneficial to be able to test the entire satellite as part of HIL, and our team attempted to get close to this with the use of a remote FlatSat, a local FlatSat (i.e., the LidSat), and partial reconstructions of the expected CubeSat system.

The use of automated testing, i.e. CI, test suites, and the Jenkins server that connects to P-HIL, lowers the barriers to perform testing and ensures repeatability in testing procedures. We can perform regression and acceptance testing for the satellite payload within this test configuration. Setting up this test configuration comes at an initial cost, but makes it easier to perform consistent tests repeatedly. Setting it up took two months. Thus the overall cost of setting up the test configuration was justified. Adding more testing functionality became more straightforward with this setup and establishing the protocols for creating tests.

Manual testing is labor-intensive but has some clear advantages. The tester can determine if the code performs as intended, and we have experienced that a different perspective from a third party has proven fruitful. The process has also been used to request essential changes, both for UI and for operations, and more parts of the team became more familiar with the codebase by doing manual testing. It isn't easy to quantify the amount of manual testing versus automatic testing that is performed, but more effort is definitely put into manual testing. During manual testing, we have also started to do these sessions using a *Remote Mob Programming*-strategy, where one *driver* is responsible for sharing their screen and conducting code changes, whilst the other team members provide helpful comments, questions, and perspectives.

The most labor-intensive testing performed was the rehearsals, but they also provided instrumental insight in terms of operations. Over the development period, we conducted two such rehearsals. During rehearsals, where communication outages were simulated, it became clear that some of the needed functionality for operations had not yet been properly specified. This testing helped us specify and develop that functionality while at the same time uncovering completely new problems that would not be caught with automated testing or code reviews.

The rigorous and extensive testing strategies used by large space organizations, such as ESA are not always feasible for a small CubeSat team, and they do not guarantee mission success [275]. The testing strategies given here are not as rigorous or as extensive as they could be but can prove adequate for the size of the team and the accepted risk. The rehearsals and the different HIL-like setups lead to a firm foundation upon which to build further tests.

11.5 Conclusions

In this chapter, we have described the testing strategies adopted by the HYPSO team in the development of their first satellite, presented the issues that these strategies uncovered, and discussed the benefits and limitations of these testing strategies.

Certain aspects of the mission have not been tested end-to-end on target hardware before launch. The planned image processing pipelines are still under development at launch and need more testing. These processing pipelines will extend the capabilities of the payload and are intended to utilize and provide relevant information from and to other assets [23, 268].

The time referencing of the existing system has not been properly tested as the available clock hardware has not reflected the actual target hardware for this part of the system. This is a consequence of relying on a remote FlatSat.

Early testing is essential, as it permits more time to resolve issues. Having a testing infrastructure available for as many developers and reviewers as possible has proven very useful. The automated testing setups and the nominal test suites provide good test repeatability and can be extended with little additional overhead. The rehearsals and manual testing during reviews have helped discover

other problems that were not caught by automated testing. Having an operator in the loop when designing software is deemed crucial for mission success, in addition to automated testing.

ACKNOWLEDGMENT

The authors would like to thank all the MSc. students contributing to testing, especially Esten S. Dalseg. The authors also want to thank Amund Gjersvik for his contribution by providing hardware, infrastructure and solving engineering problems related to making the hardware available for testing and E.F. Prentice for collaboration on the environmental tests. Finally, we are grateful for the participation of the CubeSat vendor NanoAvionics and their continuous feedback throughout the project.

Chapter 12

Discussion and Conclusions

I may not have gone where I intended to go,
but I think I have ended up where I needed to be.

Douglas Adams
The Long Dark Tea-Time of the Soul

12.1 Addressing the Research Questions

Chapter 1 introduced the following research questions:

- **RQ-1.** How can a CubeSat platform be developed for hyperspectral ocean color applications and provide end-users with valuable information?
- **RQ-2.** What kind of algorithms or models for better hyperspectral data acquisition and reduced data latency can be deployed on-board a CubeSat tailored for ocean color observations?
- **RQ-3.** How can a CubeSat be used to support the development, validation, and upgradeability of new in-orbit data processing algorithms?

This concluding chapter aims to discuss how the contributions of the Ph.D. research presented here attempt to answer these questions.

Chapter 4 presents some of the limitations of our current understanding of bio-optical properties in optically complex waters exemplified by the Amazon River Plume. Even though the results are shown in their preliminary form, it is possible to conclude that algorithms to derive IOPs work well for some conditions but are not able to capture all aspects of these water types. Expanding the QAA to better infer the properties of the spectral slope S for the computation of absorption related to CDOM seems like the most fitting path to investigate further, given the data as it is presented here.

Moreover, with equipment, such as the radiometer used in 4, it should be a practical approach to accelerate algorithm development for ocean color data. This combination of instrumentation platforms or autonomous assets of some kind would enable a coupled data acquisition and interpretation for selected areas. This concept is, to some extent, the vision of the MASSIVE project.

Thus, this system of systems should be able to effectively monitor areas of particular interest. This system can aid in targetting the sixth sustainability goal related to the availability and management of drinking water and sanitation. If the whole system is deployed in drinking water reservoirs, such as Maridalsvannet outside of Oslo [276], this could help monitor the vital resource for the largest city in Norway. Furthermore, the approach can be used by governmental agencies or other interested parties to monitor areas of high human activity where the focus would be the fourteenth sustainability goal related to conservation and use of the oceans, seas, and marine resources.

12.1.1 A CubeSat platform for Ocean Color Observations

Chapter 2, with appendix A, presents a CubeSat platform with hyperspectral imaging capabilities and accompanying processing strategies to reduce the data latency and deliver ocean color data products by using appropriate on-board processing. With the camera characteristics described in [5], the system should be able to provide data of similar quality as other more traditional Earth observation platforms.

Fitting these capabilities into a 6U CubeSat platform shows a potential disruptive path of performing hyperspectral Earth observation, as it can provide a more cost-effective solution to resolve some of the same problems. The concept and design that the payload is based on are presented in [69]. This design aims to produce hyperspectral images capable of capturing hyperspectral data cubes of relatively high quality in a miniaturized instrument suitable for deploying on smaller platforms cost-effectively.

How our team extended this design to accommodate better the specific needs of hyperspectral ocean color remote sensing is described in [5, 116]. The resulting payload deployed on HYPSO-1 has a total mass of approximately 1.3 kg, not including the electronics stack. The HSI payload is thus of a similar size and weight as other miniaturized hyperspectral imagers for cuebsats [5], and has performance metrics previously associated with larger, more conventional hyperspectral Earth observation payloads. For comparison, the HICO mission consisted of a hyperspectral imager weighing more than 40 kilos, could not control its attitude freely, and was controlled by a Windows OS that ceased to function regurarly [127]. The HICO mission level success is a topic for some debate, but it is difficult to dispute that it has provided a data record of hyperspectral images of coastal regions that have proven valuable for developing future missions. The camera performance metrics associated with the HICO instrument are similar to the simulated and tested performance found for the HYPSO-1 payload [5, 127]. Our payload thus shows how a hyperspectral imager for ocean color can be developed for a CubeSat platform.

The limitations due to communication links will pose a challenge when delivering end-users with valuable information. However, these limitations will be mitigated by appropriate on-board processing in the HYPSO-1 system. In addition to the hyperspectral imager, the payload computer dubbed OPU controls both the RGB and HSI. Chapter 10 presents details on software architecture and other relevant design choices aimed at delivering end-users with valuable information for the OPU. The OPU is equipped with a FPGA and CPU and is running a well-tested Linux-based OS specified by the SoC manufacturer. The testing regiment described in chapter 11 has shown that the OPU can perform the expected nominal operations without any unexpected artifacts in the expected thermal-vacuum environment where it will operate. These nominal operations, in addition to capturing RGB and HSI images, include routine software updates of parts of the system. The OPU can perform power-cycling and has no persistence in the OS; This, among other factors, makes it possible to change the behavior of the system at an acceptable level of risk. Changing the system's behavior makes it possible to tailor the data acquisition to changing end-user specifications even after launch.

12.1.2 Algorithms and Models for efficient Ocean Observations

As has been emphasized throughout this thesis, hyperspectral imagers can quickly acquire large amounts of data. For spaceborne platforms, such as HYPSO-1, one of the major bottlenecks is challenges concerning the bandwidth of the available communication links. In chapter 2 it is stated that the HYPSO-1 satellite will have an S-band Transceiver. This radio provides a usable data rate of 1 Mbps for down-linking, which is not that much when considering that a single raw data cube is estimated to be in the range of 150 MBs for nominal operations.

Given that the ocean can be a highly dynamic environment, the data latency will affect the utility of the collected data for some use cases. The on-board processing will need to be adaptable to cater to multiple end-user applications. In some cases, end-users will appreciate lossless data and, in other cases, data without any significant information loss. This trade-off analysis for data acquisition becomes relevant when conducting a longitudinal study of a given area with multiple different methods and platforms for ocean observation that are not permanent installations. On the other hand, more operational considerations should be accounted for when providing information for fisheries management and other decision-makers where timeliness is essential. Exploring the data and what kind of insights it can provide is not the main focus in these more operational scenarios. Thus the end-user should be able to accept a lossy data retrieval. That is, as long as the retrieved information is still adequately accurate and delivered with an acceptable latency.

In chapter 5, 6, 7, and 8, this thesis provides some strategies and perspectives on how to reduce the data latency. These are not tested on the target hardware. However, they represent simplistic models suited to be deployed on the target hardware.

The chapters 5 and 6 focus on more general hyperspectral image processing. The former chapter focuses on using dimensionality reduction to provide a subspace that can be analyzed by more

exploratory algorithms, here exemplified by target detection. This chapter demonstrates how to mitigate the limitations of transformation-based encoding for HSI compression by separating the transformed spectra into a known subspace using a pre-computed transformation matrix for dimensionality reduction and residuals. Thus, the reduced subspace can provide similar performance in terms of target detection as the original spectral space but with a reduced number of floating-point operations required to compute the result. The subspace is expected to perform similarly when used with other algorithms attempting to retrieve information from the entire spectral space available, such as classification. How this subspace can be computed at a predetermined bitrate is discussed in chapter 6. The subspace can be used as discussed in chapter 5 or be used for further source encoding for HSI compression. The computed residuals can be used to improve the pre-computed transformation matrix, provide insight into the limitations of the transformation, and provide greater confidence in data that is retrieved using transformation-based encoding as a lossy encoding scheme.

Chapter 7 describes methods for AC using both NN and PLS. As briefly discussed here and detailed in [199], both these modeling strategies were able to provide the reported performance on the simulated data set with relatively simplistic models. These models can be deployed on the target hardware when combined with appropriate pre-processing. Either on the CPU, or the FPGA, and this should make it possible to derive the parameters of interest in-orbit. With AC, where the viewing geometry and atmospheric conditions are corrected for, the data will be ready for other processing steps. By doing the AC as part of the pre-processing, the later processing stages will get consistent spectra for the same or similar conditions, given that the AC works. This could be especially important when classifying or detecting a priori spectra through classification or target detection.

This shows that models could compensate for the viewing geometry and directly derive desirable ocean color parameters. As detailed in chapter 2, this could provide end-users with the relevant data more rapidly than if the entire data cube were to be downloaded. Furthermore, chapter 7 also reports how these modeling strategies would perform when attempting to retrieve IOPs or end-user specified data products derived from these parameters [7] using the simulated data with the same simple model strategies. These options are, of course, not mutually exclusive, i.e., operationally HYPSO-1 could provide both types of data for different end-users for the same area.

Chapter 8 provides a different validation strategy when testing similar models for similar purposes, as shown in chapter 7. Here real-world data sets from the HICO mission are used and processed through SeaDAS 7.2. The end products, after standard pre-processing and AC by SeaDAS 7.2, are compared with simplistic pre-processing and simple linear models using ToA radiance to ocean color parameters of interest. The results show that using a straightforward strategy for correcting the viewing geometry and allowing the models to infer the ocean color parameters of interest directly from ToA can give acceptable results when compared to the accuracy of the OC4 algorithm [16] when using data processed by SeaDAS 7.2 as a baseline.

Thus, these simplistic models shown in chapter 7 and 8 could be used to process the data on-board and provide an indication of the [Chl a] of a given area. As mentioned in [7, 16], band-ratio approaches do give a reliable relative measure of the [Chl a], but not necessarily an excellent quantitative measure. This means that even if one were to down-link the raw data and rely on band-ratio [Chl a] estimation algorithms, one would at best get a good relative measure of the concentration. From a more operational perspective, it could be beneficial to reduce the data latency by accepting the (relative) [Chl a] estimation from a different model that uses only the ToA and viewing geometry as input and can be performed as part of the on-board processing. By reducing the latency, this way, a map of the relative [Chl a] could be used to guide the other autonomous assets discussed in Chap 2 and 3 as well as in [24], with reduced latency. The data that these assets can collect can then be used to adjust the models used on-board through in-flight software updates.

12.1.3 Adaptable Software for a CubeSat Payload

NTNU, as a research institution, wants to develop competence regarding the use of CubeSat technology. This research includes investigating how to best utilize different payloads on CubeSat systems. With this goal, there are clear benefits, not limited to research, of enabling the development, validation, and upgrade-ability of new data processing algorithms in-orbit for these payloads. For example, hyperspectral sensors from space for ocean color is a less explored scientific topic; it will be convenient to adjust the algorithms used on-board during its operational phase.

Doing software upgrades in-orbit poses some risk; one can not be sure that the system is altered in a way so that it will not cease to function. However, enabling in-orbit upgrades would extend the payload's capabilities beyond its initial launch configuration. This thesis presents a strategy for the software architecture design and testing of the payload that enables this functionality. The concept described in chapter 2 deems the risk of in-orbit upgrades for HYPSO-1 to be at an acceptable level for the software architecture used.

By using a distributed service-oriented architecture as briefly described in chapter 10, with more details given in [259, 260, 267], HYPSO-1 is expected to be able to do in-orbit upgrades. This functionality is made possible by running a well-established Linux-based OS, and making sure that the payload handler is running on top of that system in the application layer. Thus, one can upload different versions of the application that handles the complex payload operations to OPU. This file or executable running in the application layer supports changing between different versions of itself at run-time. Consequently, this makes it possible to have multiple application versions available. Changing between them does pose a negligible risk, and if a given application ends up crashing the OS unexpectedly, there is always the option of performing a power-cycle of the OPU, as the subsystem for power control, the EPS, is entirely separate and again is controlled by other subsystems.

The risks of doing in-orbit upgrades are thus significantly reduced due to no persistence in the OS in addition to a predefined booting sequence that will start the application that the OS image was packaged with initially. The software design and implementation presented here is one approach to support the development, validation, and upgrade-ability of new data processing algorithms in-orbit, to some extent independent of payload instrument or sensor.

12.1.4 Research Limitations

As stated in the introduction;

"This thesis documents an introduction to some challenges of hyperspectral remote sensing for ocean color observations, CubeSat development, and Software development. This is not an extensive exploration of all there is to know about these complex topics but is intended to give insight into some of the considerations taken into account when developing HYPSO-1."

It has been challenging to determine what kind of research directions were appropriate given that the HYPSO-1 platform and the presented methods were under development during the entire research period. A significant amount of time and energy during the Ph.D. has been focused on leading the development of the OPU and associated payload software. As a result, the OPU design, to some extent, supports changes in requirements or ways we want to use the hyperspectral payload in-flight. When comparing the learning outcomes of developing this system as a team project to using the Ph.D. research period more focused, it is difficult to say what the different results could have been. It is possible that the individual researcher, being me, could have made more contributions to some of the research fields discussed here, and the impact of these contributions would be purely speculative. However, the learning outcomes that affect the students affiliated with the project are evident. As the system development was done as a team project, this presented a venue for sharing experiences and knowledge. In other words, the learning outcomes for the individual might have been reduced for the added benefit of accelerating the learning outcomes of the group. The learning outcomes of the individual might also have been broadened and even accelerated by the collaboration.

It is with great pleasure that I can conclude this thesis by saying that the HYPSO-1 system has been launched into space successfully. It was launched as part of a rideshare mission by SpaceX on the 13th of January, along with 105 other satellites. When completing this thesis, our team is still commissioning the satellite. So far, the payload and other subsystems are behaving as expected. I had the pleasure of pinging the payload for the first time on the 28th of January. We expect the HYPSO-1 system to be able to investigate further the research questions raised in this thesis.

12.2 Future Perspectives

This section provides some perspectives and hopes regarding how the HYPSO-1 system and accompanying MASSIVE project, and proposed algorithms can provide helpful information and lessons learned for potential future CubeSat developers and data product end-users.

12.2.1 Future Satellite Development

The HYPSO team has started the development of their second CubeSat that will have an upgraded version of the hyperspectral payload, increased processing capabilities, and a SDR [277]. Based on the experiences from HYPSO-1, the team plans to continue the agile work methodology for both hardware and software, and increase the importance of team building and team cohesiveness, as described in chapter 9. The team is also considering introducing MVPs and a clearer "definitions of done" [237, 278], which could increase the sprint performance. More extensive specifications before implementation will also be attempted for software development. All these lessons learned will help improve HYPSO-2, the next planned CubeSat mission.

The team has introduced a cloud-based digital tool for managing requirements, system budgets, analysis, verification planning, and project planning. Previously, this effort was managed through the systems engineer, but now, the team can collaborate in real-time from different sites on the same set of requirements. These updates also feed automatically into system budgets and the product breakdown structure. The team members can create discussions, flag components or requirements, and assign tasks within the team. This is a part of the "Central, shared, digital information system" shown in Figure 12.1.

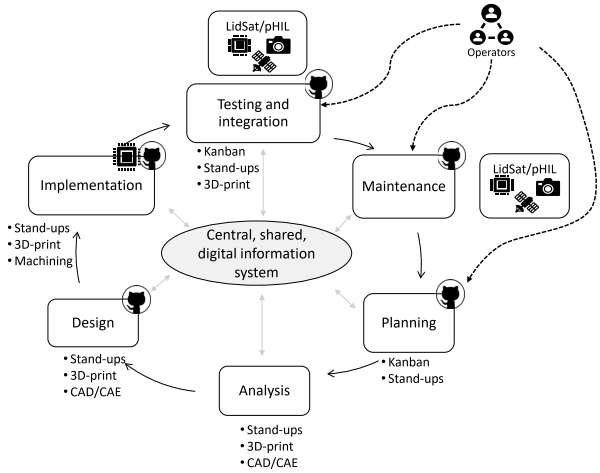


Figure 12.1.: Product development lifecycle with digital engineering methods and tools. Figure courtesy of Evelyn Honoré-Livermore

12.2.2 On-Board Processing of Hyperspectral Ocean Color Data

It can be challenging to determine what products one should archive for future data records. In [10] it is shown one example of how data products from different ocean color satellites and other augmenting sampling platforms can be used in unison to provide insight on long-term climate changes over 20 years. One silver lining of the results presented in [10] is that they do not use the raw data products from the ocean color satellites but instead derived data products. One of the goals of the HYPSO project is to derive ocean color products of interest as part of on-board processing. These derived data products would need quality parameters associated with them to be relevant for policy makers [2]. How to extend the proposed algorithmic approaches intended to effectively derive ocean color data products on-board HYPSO-1, as well as including other relevant parameters, such as quality parameters, is a topic for future research.

12.3 Observational Pyramid

This thesis describes the work performed to create the top of the observational pyramid envisioned by the MASSIVE project. Namely a CubeSat with its HSI payload.

The intention of the CubeSat platform described in this thesis is not to cover the same global spatial scales that traditional Earth observation satellites do. It aims at tailoring the data acquisition for given areas. The CubeSat and its payloads should deliver relevant and valuable information to a given end-user by deploying apt on-board processing.

Given that the system will target smaller areas, it is appropriate to support the calibration and validation activities of the acquired data using other autonomous assets deployed in the same places. When the system of systems becomes more integrated, the data acquisition will function as a feedback loop to control and correct the expected degradation and altercation of the electro-optical components of the HSI payload. Furthermore, this feedback property can produce better and more accurate models for a given area over time.

The envisioned observational pyramid, depicted in Figure 1.4, provides a modern and cost-effective infrastructure concept to better monitor ocean and water areas of interest. By implementing the payload of the apex of the observational pyramid as proposed in this thesis, one could conceivably reach this vision.

Appendix A

HYPSO-1 System

A.1 Satellite Bus

The hyperspectral imager was chosen to be adapted to the Multipurpose 6U Platform (M6P), a commercially available spacecraft bus provided by NanoAvionics, with a mass of approximately 6.8 kg when fully integrated. Among the crucial subsystems in M6P are Flight Computer (FC) for on-board data handling and ADCS functions, SatLab Global Navigation Satellite System (GNSS) for orbit determination and time synchronization, Electrical Power System (EPS), Ultra-High-Frequency (UHF) radio for basic space-ground communications, and Payload Controller (PC) working as storage device and router between the payload and the satellite bus. For internal communications, the spacecraft uses the CubeSat Space Protocol (CSP) over a Controller Area Network (CAN), where each subsystem is a network node with a dedicated CSP address. The M6P has 16 body-mounted triple junctions Gallium Arsenide solar cells and six Lithium-Ion batteries with a total energy capacity of 64.9 Wh (As of 2020).

A.2 Other Components and Subsystems

To fulfill the user needs and mission CONOPS described in Section 2.1, HYPSO-1 is further equipped with:

- A Nano Star Tracker ST-1 [279], and Sensor STIM 210 Inertial Measurement Unit (IMU) [280] used for precise attitude estimation during imaging. To ensure sufficient settling time after initialization, the sensors are turned on for at least 5 min prior to imaging. When images are not taken, then six sun sensors, three magnetometers, and three gyroscopes are used instead, which provide coarser attitude knowledge but consume less power;
- Four reaction wheels used for attitude control that provide up to 3.2 mNm torque each, where three are placed orthogonally along the body axes and the fourth is tilted at an angle of $54.7°$. Two magnetorquers are placed along each body axis for reaction wheel momentum dumping;

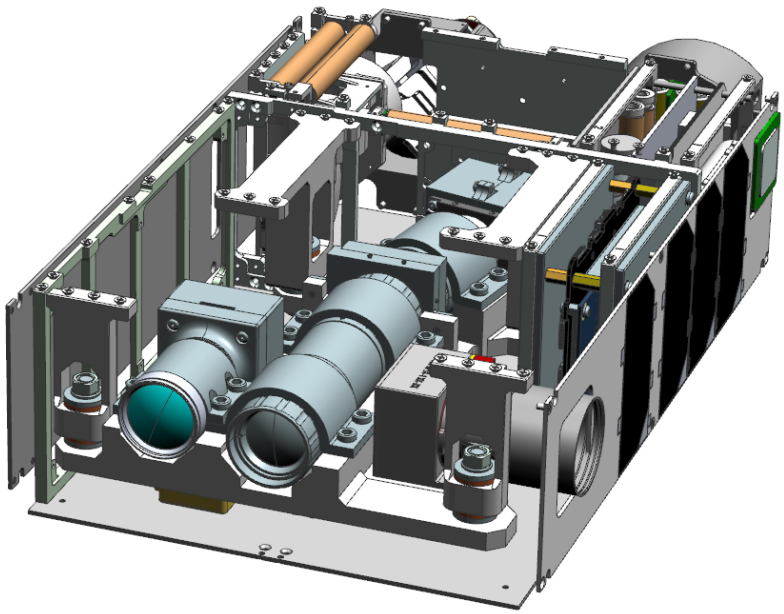


Figure A.1.: Computer-Aided Drawing (CAD) model of HYPSO-1 with its top and front panels removed showing the hyperspectral imager in the center, RGB camera to its left and star-tracker to its right. Provided by Elizabeth F. Prentice

- An IDS UI-125x RGB camera with 6 mm $F/1.4$ Ci series fixed lens providing a footprint of 770 km $\times$ 540 km and spatial resolution of approximately 500 m. Its main purpose is to support and validate hyperspectral images in the spatial domain [281];
- A 2.4 GHz IQ Spacecom S-band Transceiver providing usable data rate of 1 Mbps for downlinking payload data;
- An On-board Processing Unit (OPU) hosting a Zynq-7030 Xilinx PicoZed System-on-a-Chip (SoC) with flight heritage [53]. It consists of two core ARM processors and a Field Programmable Gate Array (FPGA) dedicated for on-board image processing. The OPU allows for in-orbit updates of both software and FPGA hardware reconfigurations for uploaded algorithms. Larger data sizes can be buffered from the OPU to the PC over CAN before downlinking over S-band radio, or smaller amounts of data can be downloaded directly from the OPU. Buffering data to the PC enables full utilization of the S-band data rate, and removes the need for keeping the OPU turned on for longer than necessary. Power and data-line distribution to the hyperspectral and RGB cameras are granted through a custom break-out board with PicoZed interfaces. Furthermore, the OPU hosts a SD-card with 8 GB storage capacity.

A.3 Power Budget

M6P's solar arrays generate approximately 11.65 W during a period of 58.9 min in sunlight out of a total orbital period of 94.6 min. Determining if energy is sufficient during burdensome operations, the power budget should assume a scenario where image acquisition, processing, and downlink all happen in the same pass during sunlight. This scenario is shown in Figure A.2 for HYPSO-1 passing over a target area in Lofoten, Norway, and the selected ground stations at NTNU Trondheim, KSAT Svalbard, and KSAT Spain.

Table A.1.: HYPSO-1 Power Budget

Subsystem	Power (W)	DC (%)	Power Used (W)
Hyperspectral imager	3.675	1.09	0.040
RGB camera	3.375	0.55	0.020
OPU imaging	4.234	1.09	0.046
OPU image processing	4.234	6.69	0.283
OPU-PC transfer	4.234	35.33	1.496
ADCS cruise	3.441	94.72	3.259
ADCS precise	6.331	5.28	0.334
S-band radio RX	4.813	10.57	0.509
S-band radio TX+RX	12.201	10.57	1.290
Other	1.530	100	1.530
Total (+10% margin)			9.688
Generated (effective)			9.861
Remaining			+0.174

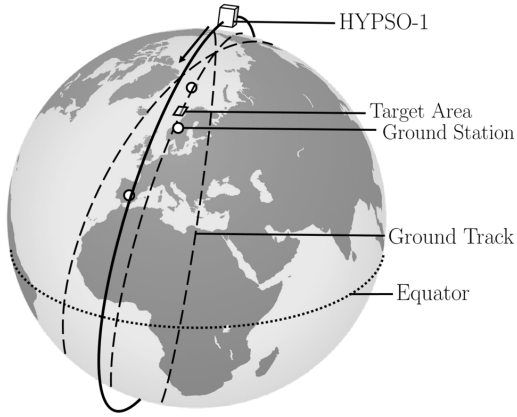


Figure A.2.: HYPSO-1 in SSO at 10:25:00 on 28 May 2022. Selected ground stations are marked in white circles. Previous, current and succeeding ground tracks are indicated by dashed lines.

Table A.1 shows the power budget with 5% component margin and the corresponding subsystem duty cycles (DC) that includes booting up. Battery input and output efficiencies are assumed 92% each. The power consumed in OPU, ADCS and S-band radio are separated into more than one operational mode, while "Other" denotes the collective power consumption by FC, EPS, PC and internal bus communications. Naturally, peaks in power are expected during image acquisition, image processing and downlink. "ADCS precise" indicates preparing and executing the slew maneuver during image acquisition when both the IMU and star-tracker are active, consuming up to 1.5 W each. Adding a 10% system margin results in remaining power of about 174 mW. Enforcing the power budget to remain safe and positive, the allowed duration is set to maximum 6.33 min for onboard image processing and 33.42 min for transferring data from OPU to PC through CAN. Allowed duration of data transmission through S-band radio is set to 10 min which enables downlinking up to 75 MB of data per orbit.

Bibliography

[1] Heidi M. Dierssen, Steven G. Ackleson, Karen E. Joyce, Erin L. Hestir, Alexandre Castagna, Samantha Lavender, and Margaret A. McManus. Living up to the hype of hyperspectral aquatic remote sensing: Science, resources and outlook. *Frontiers in Environmental Science*, 9:134, 2021.

[2] T. Platt, N. Hoepffner, V. Stuart, and C. Brown. *Why Ocean Colour? The Societal Benefits of Ocean-Colour Technology*, volume No. 7 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2008.

[3] United Nations. Goal 14 | department of economic and social affairs. `https://sdgs.un.org/goals/`. (Accessed on 11/05/2021).

[4] M. Eismann. *Hyperspectral remote sensing*. Society of Photo-Optical Instrumentation Engineers, 2012.

[5] Elizabeth Frances Prentice, Mariusz Eivind Grøtte, Fred Sigernes, and Tor Arne Johansen. Design of a hyperspectral imager using COTS optics for small satellite applications. In Bruno Cugny, Zoran Sodnik, and Nikos Karafolas, editors, *International Conference on Space Optics — ICSO 2020*, volume 11852, pages 2172 – 2189. International Society for Optics and Photonics, SPIE, 2021.

[6] Zsolt Volent, Geir Johnsen, and Fred Sigernes. Kelp forest mapping by use of airborne hyperspectral imager. *Journal of Applied Remote Sensing*, 1(1):1 – 21, 2007.

[7] Z-P. Lee. *Remote Sensing of Inherent Optical Properties: Fundamentals, Tests of Algorithms, and Applications*, volume No. 5 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2006.

[8] P. Jeremy Werdell, Michael J. Behrenfeld, Paula S. Bontempi, Emmanuel Boss, Brian Cairns, Gary T. Davis, Bryan A. Franz, Ulrik B. Gliese, Eric T. Gorman, Otto Hasekamp, Kirk D. Knobelspiesse, Antonio Mannino, J. Vanderlei Martins, Charles R. McClain, Gerhard Meister, and Lorraine A. Remer. The Plankton, Aerosol, Cloud, Ocean Ecosystem Mission: Status, Science, Advances. *Bull. Am. Meteorol. Soc.*, 100(9):1775–1794, 2019.

[9] M. Wang. *Atmospheric Correction for Remotely-Sensed Ocean-Colour Products*, volume No. 10 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2010.

[10] Edson Silva, François Counillon, Julien Brajard, Anton Korosov, Lasse H. Pettersson, Annette Samuelsen, and Noel Keenlyside. Twenty-one years of phytoplankton bloom phenology in the barents, norwegian, and north seas. *Frontiers in Marine Science*, 8:1626, 2021.

[11] Zsolt Volent, Geir Johnsen, Erlend K. Hovland, Are Folkestad, Lasse M. Olsen, Karl Tangen, and Kai Sorensen. Improved monitoring of phytoplankton bloom dynamics in a Norwegian fjord by integrating satellite data, pigment analysis, and Ferrybox data with a coastal observation network. *Journal of Applied Remote Sensing*, 5(1):1 – 22, 2011.

[12] Norges kontinentalsokkel – store norske leksikon. `https://snl.no/Norges_kontinentalsokkel`. (Accessed on 10/04/2021).

[13] G. Johnsen, K. Tangen Z. Volent, and E. Sakshaug. Time series of harmful and benign phytoplanktonblooms in northwest European waters using the Seawatch buoy system. In *Monitoring Algal Blooms: New Technologies for Detecting Large-Scale Environmental Change*, chapter 6, pages 113–141. Springer-Verlag and Landes Bioscience, 1997.

[14] Amir Ibrahim, Bryan Franz, Ziauddin Ahmad, Richard Healy, Kirk Knobelspiesse, Bo-Cai Gao, Chris Proctor, and Peng-Wang Zhai. Atmospheric correction for hyperspectral ocean color retrieval with application to the hyperspectral imager for the coastal ocean (hico). *Remote Sensing of Environment*, 204:60–75, January 2018.

[15] Catherine Kuhn, Aline de Matos Valerio, Nick Ward, Luke Loken, Henrique Oliveira Sawakuchi, Milton Kampel, Jeffrey Richey, Philipp Stadler, John Crawford, Rob Striegl, Eric Vermote, Nima Pahlevan, and David Butman. Performance of landsat-8 and sentinel-2 surface reflectance products for river remote sensing retrievals of chlorophyll-a and turbidity. *Remote Sensing of Environment*, 224:104–118, 2019.

[16] John E O'Reilly and P Jeremy Werdell. Chlorophyll algorithms for ocean color sensors-oc4, oc5 & oc6. *Remote sensing of environment*, 229:32–47, 2019.

[17] Seelye Martin. *Ocean color*, page 136–193. Cambridge University Press, 2 edition, 2014.

[18] Erik Kulu. Nanosats database. online, 2020. URL: `https://www.nanosats.eu`, Accessed 2020-12-11.

[19] Jordi Puig-Suari, Clark Turner, and William Ahlgren. Development of the standard cubesat deployer and a cubesat class picosatellite. In *2001 IEEE aerospace conference proceedings (Cat. No. 01TH8542)*, volume 1, pages 1–347. IEEE, 2001. `https://doi.org/10.1109/AERO.2001.931726`.

[20] Armen Poghosyan and Alessandro Golkar. CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions. *Progress in Aerospace Sciences*, 88:59–83, January 2017.

[21] M. Rizwan Mughal, J. Praks, R. Vainio, P. Janhunen, J. Envall, A. Näsilä, P. Oleynik, P. Niemelä, S. Nyman, A. Slavinskis, J. Gieseler, N. Jovanovic, B. Riwanto, P. Toivanen, H. Leppinen, T. Tikka, A. Punkkinen, R. Punkkinen, H.-P. Hedman, J.-O. Lill, and J.M.K. Slotte. Aalto-1, multi-payload cubesat: In-orbit results and lessons learned. *Acta Astronautica*, 187:557–568, 2021.

[22] Lucinda Berthoud, Michael Swartout, James Cutler, David Klumpar, Jesper A. Larsen, and Jens Dalsgaard Nielsen. University cubesat project management for success. In *Proceedings of the Conference on Small Satellites*, 2019. `https://digitalcommons.usu.edu/smallsat/2019/all2019/63/`.

[23] 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*, pages 1–19, 2021.

[24] Alberto Dallolio, Gara Quintana-Diaz, Evelyn Honoré-Livermore, Joseph L. Garrett, Roger Birkeland, and Tor A. Johansen. A satellite-usv system for persistent observation of mesoscale oceanographic phenomena. *Remote Sensing*, 13(16), 2021.

[25] Geir Johnsen, Mark A. Moline, Lasse H. Pettersson, James Pinckney, Dimitry V. Pozdnayakov, Einar S. Egeland, and Oscar M. Schofield. Optical monitoring of phytoplankton bloom pigment signatures. In S. Roy, C. A. Llewellyn, E. S. Egeland, and G. Johnsen, editors, *Phytoplankton Pigments: Characterization, Chemotaxonomy and Applications in Oceanography*, chapter 14, pages 538–606. Cambridge University Press, 2011.

[26] S. Sathyendranath. *Phytoplankton Functional Types from Space*, volume No. 15 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2014.

[27] David Blondeau-Patissier, James F.R. Gower, Arnold G. Dekker, Stuart R. Phinn, and Vittorio E. Brando. A review of ocean color remote sensing methods and statistical techniques for the detection, mapping and analysis of phytoplankton blooms in coastal and open oceans. *Prog. Oceanogr.*, 123:123–144, 2014.

[28] T. Kutser, L. Metsamaa, N. Strombeck, and E. Vahtmae. Monitoring cyanobacterial blooms by satellite remote sensing. *Estuar. Coast. Shelf Sci.*, 67:303–312, 2006.

[29] G. Johnsen, O. Samset, L. Granskog, and E. Sakshaug. In vivo absorption characteristics in 10 classes of bloom-forming phytoplankton: taxonomic characteristics and responses to photoadaptation by means of discriminant and HPLC analysis. *Mar. Ecol. Prog. Ser.*,

105:149–157, 1994.

[30] David A. Jessup, Melissa A. Miller, John P. Ryan, Hannah M. Nevins, Heather A. Kerkering, Abdou Mekebri, David B. Crane, Tyler A. Johnson, and Raphael M. Kudela. Mass Stranding of Marine Birds Caused by a Surfactant-Producing Red Tide. *PLoS One*, 4(2):1–8, 2009.

[31] T. D. Dickey and R. R. Bidigare. Interdisciplinary oceanographic observations: the wave of the future. *Sci. Mar.*, 69:23–42, 2005.

[32] R. Frouin. *In-flight Calibration of Satellite Ocean-Colour Sensors*, volume No. 14 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2013.

[33] M-H. Forget, V. Stuart, and T. Platt. *Remote Sensing in Fisheries and Aquaculture*, volume No. 8 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2009.

[34] Curtiss O. Davis, Jeffrey Bowles, Robert A. Leathers, Dan Korwan, T. Valerie Downes, William A. Snyder, W. Joe Rhea, Wei Chen, John Fisher, W. Paul Bissett, and Robert Alan Reisse. Ocean PHILLS hyperspectral imager: design, characterization, and calibration. *Opt. Express*, 10(4):210–221, 2002.

[35] M. R. Corson, D. R. Korwan, R. L. Lucke, W. A. Snyder, and C. O. Davis. The Hyperspectral Imager for the Coastal Ocean (HICO) on the International Space Station. In *IGARSS 2008 - 2008 IEEE Int. Geosci. Remote Sens. Symp.*, volume 4, pages 101–104, July 2008.

[36] Bo-Cai Gao, Marcos J. Montes, Curtiss O. Davis, and Alexander F.H. Goetz. Atmospheric correction algorithms for hyperspectral remote sensing data of land and ocean. *Remote Sensing of Environment*, 113:S17–S24, September 2009.

[37] Edward J. Knight and Geir Kvaran. Landsat-8 Operational Land Imager Design, Characterization and Performance. *Remote Sens.*, 6(11):10286–10305, 2014.

[38] Miguel Aguirre, Bruno Berruti, Jean-Loup Bezy, Mark Drinkwater, Florence Heliere, Ulf Klein, Constantinos Mavrocordatos, Pierluigi Silvestrin, Bruno Greco, and Jerome Benveniste. Sentinel-3 - the ocean and medium-resolution land mission for GMES operational services. *ESA Bull.*, 131:24–29, Aug 2007.

[39] Frank E. Muller-Karger, Erin Hestir, Christiana Ade, Kevin Turpie, Dar A. Roberts, David Siegel, Robert J. Miller, David Humm, Noam Izenberg, Mary Keller, Frank Morgan, Robert Frouin, Arnold G. Dekker, Royal Gardner, James Goodman, Blake Schaeffer, Bryan A. Franz, Nima Pahlevan, Antonio G. Mannino, Javier A. Concha, Steven G. Ackleson, Kyle C. Cavanaugh, Anastasia Romanou, Maria Tzortziou, Emmanuel S. Boss, Ryan Pavlick, Anthony Freeman, Cecile S. Rousseaux, John Dunne, Matthew C. Long, Eduardo Klein, Galen A. McKinley, Joachim Goes, Ricardo Letelier, Maria Kavanaugh, Mitchell Roffer, Astrid Bracher, Kevin R. Arrigo, Heidi Dierssen, Xiaodong Zhang, Frank W. Davis, Ben

Best, Robert Guralnick, John Moisan, Heidi M. Sosik, Raphael Kudela, Colleen B. Mouw, Andrew H. Barnard, Sherry Palacios, Collin Roesler, Evangelia G. Drakou, Ward Appeltans, and Walter Jetz. Satellite sensor requirements for monitoring essential biodiversity variables of coastal ecosystems. *Ecol. Appl.*, 28(3):749–760, 2018.

[40] Igor Ogashawara. The Use of Sentinel-3 Imagery to Monitor Cyanobacterial Blooms. *Environments*, 6(6, 60), 2019.

[41] Fred Ortenberg. *Hyperspectral Sensor Characteristics*, pages 39–68. Hyperspectral Remote Sensing of Vegetation, 2011.

[42] Jennifer L. Wolny, Michelle C. Tomlinson, Stephanie Schollaert Uz, Todd A. Egerton, John R. McKay, Andrew Meredith, Kimberly S. Reece, Gail P. Scott, and Richard P. Stumpf. Current and Future Remote Sensing of Harmful Algal Blooms in the Chesapeake Bay to Support the Shellfish Industry. *Front. Mar. Sci.*, 7, 2020.

[43] M. Soukup, J. Gailas, D. Fantin, A. Jochemsen, C. Aas, P. J. Baeck, L. Benhadj, S. Livens, B. Delauré, M. Menenti, B. G. H. Gorte, S. E. Aria Hosseini, M. Esposito, and C. N. van Dijk. HyperScout: Onboard Processing of Hyperspectral Imaging Data on a Nanosatellite. In *Small Satellites, System and Services Symposium (4S), Valletta, Malta*, 2016.

[44] Heidi Dierssen, George B McManus, Adam Chlus, Dajun Qiu, Bo-Cai Gao, and Senjie Lin. Space station image captures a red tide ciliate bloom at high spectral and spatial resolution. *Proc. Natl. Acad. Sci.*, 112(48):14783–14787, 2015.

[45] Luis Guanter, Hermann Kaufmann, Karl Segl, Saskia Foerster, Christian Rogass, Sabine Chabrillat, Theres Kuester, André Hollstein, Godela Rossner, Christian Chlebek, Christoph Straif, Sebastian Fischer, Stefanie Schrader, Tobias Storch, Uta Heiden, Andreas Mueller, Martin Bachmann, Helmut Mühle, Rupert Müller, Martin Habermeyer, Andreas Ohndorf, Joachim Hill, Henning Buddenbaum, Patrick Hostert, Sebastian Van der Linden, Pedro J. Leitão, Andreas Rabe, Roland Doerffer, Hajo Krasemann, Hongyan Xi, Wolfram Mauser, Tobias Hank, Matthias Locherer, Michael Rast, Karl Staenz, and Bernhard Sang. The EnMAP Spaceborne Imaging Spectroscopy Mission for Earth Observation. *Remote Sens.*, 7(7):8830–8857, 2015.

[46] Christian Mielke, Nina Boesche, Christian Rogass, Hermann Kaufmann, Christoph Gauert, and Maarten de Wit. Spaceborne Mine Waste Mineralogy Monitoring in South Africa, Applications for Modern Push-Broom Missions: Hyperion/OLI and EnMAP/Sentinel-2. *Remote Sens.*, 6(8):6790–6816, 2014.

[47] Jay Pearlman, Pamela Barry, C. Segal, John Shepanski, Debra Beiso, and Stephen Carman. Hyperion, a Space-Based Imaging Spectrometer. *IEEE Trans. Geosci. Remote Sens.*, 41:1160 – 1173, 2003.

[48] Y. Liu, D. Sun, X. Hu, X. Ye, Y. Li, S. Liu, K. Cao, M. Chai, W. Zhou, J. Zhang, Y. Zhang, W. Sun, and L. Jiao. The Advanced Hyperspectral Imager: Aboard China's GaoFen-5 Satellite. *IEEE Geosci. Remote Sens. Mag.*, 7(4):23–32, 2019.

[49] Deren Li, Mi Wang, and Jie Jiang. China's high-resolution optical remote sensing satellites and their mapping applications. *Geo. Spat. Inf. Sci.*, 2020.

[50] Jaan Praks, Antti Kestilä, Martti Hallikainen, Heikki Saari, Jarkko Antila, Pekka Janhunen, and Rami Vainio. Aalto-1 - an experimental nanosatellite for hyperspectral remote sensing. In *2011 IEEE International Geoscience and Remote Sensing Symposium*, pages 4367–4370, 2011.

[51] A. Villafranca, J. Corbera, F. Martín, and J. F. Marchan. Limitations of Hyperspectral Earth Observation on Small Satellites. *Journal of Small Satellites*, 1(1):19–29, 2012.

[52] M Guelman and F Ortenberg. Small satellite's role in future hyperspectral Earth observation missions. *Acta Astronaut.*, 64(11):1252–1263, June 2009.

[53] A. George and C. Wilson. Onboard Processing With Hybrid and Reconfigurable Computing on Small Satellites. *Proc. IEEE*, 106:458–470, 2018.

[54] S. Lopez, T. Vladimirova, C. Gonzalez, J. Resano, D. Mozos, and A. Plaza. The Promise of Reconfigurable Computing for Hyperspectral Imaging Onboard Systems: A Review and Trends. *Proc. IEEE*, 101(3):698–722, 2013.

[55] L. Sterpone, M. Porrmann, and J. Hagemeyer. A Novel Fault Tolerant and Runtime Reconfigurable Platform for Satellite Payload Processing. *IEEE Trans. Comput.*, 62(8):1508–1525, 2013.

[56] C.-I. Chang. *Hyperspectral Imaging: Techniques for Spectral Detection and Classification*. Kluwer Academic, New York, 2003.

[57] J.C. Harsanyi and C.-I. Chang. Hyperspectral image classification and dimensionality reduction: an orthogonal subspace projection approach. *IEEE Transactions on Geoscience and Remote Sensing*, 32(4):779–785, July 1994.

[58] Raphael M. Kudela, Sherry L. Palacios, David C. Austerberry, Emma K. Accorsi, Liane S. Guild, and Juan Torres-Perez. Application of hyperspectral remote sensing to cyanobacterial blooms in inland waters. *Remote Sens. Environ.*, 167:196 – 205, 2015.

[59] Keith Davidson, Donald M. Anderson, Marcos Mateus, Beatriz Reguera, Joe Silke, Marc Sourisseau, and Julie Maguire. Forecasting the risk of harmful algal blooms. *Harmful Algae*, 53:1–7, 2016.

[60] C. R. McClain and G. Meister. *Mission Requirements for Future Ocean-Colour Sensors*, volume No. 13 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG,

Dartmouth, Canada, 2012.

[61] Lin Qi, Zhongping Lee, Chuanmin Hu, and Menghua Wang. Requirement of minimal signal-to-noise ratios of ocean color sensors and uncertainties of ocean color products. *J. Geophys. Res. Oceans*, 122(3):2595–2611, 2017.

[62] Raphael M. Kudela, Stanford B. Hooker, Henry F. Houskeeper, and Meredith McPherson. The Influence of Signal to Noise Ratio of Legacy Airborne and Satellite Sensors for Simulating Next-Generation Coastal and Inland Water Products. *Remote Sens.*, 11(18), 2019.

[63] E. Lancheros, A. Camps, H. Park, P. Sicard, A. Mangin, H. Matevosyan, and I. Lluch. Gaps Analysis and Requirements Specification for the Evolution of Copernicus System for Polar Regions Monitoring: Addressing the Challenges in the Horizon 2020–2030. *Remote Sens.*, 10(7), 2018.

[64] Curtiss O. Davis, Maria Kavanaugh, Ricardo Letelier, W. Paul Bissett, and David Kohler. Spatial and spectral resolution considerations for imaging coastal waters. In *Coastal Ocean Remote Sensing*, volume 6680, pages 196 – 207. SPIE, 2007.

[65] P. Mouroulis and R. O. Green. Review of high fidelity imaging spectrometer design for remote sensing. *Optical Engineering*, 57(4), 2018.

[66] Gregg Vane, Robert O Green, Thomas G Chrien, Harry T Enmark, Earl G Hansen, and Wallace M Porter. The airborne visible/infrared imaging spectrometer (AVIRIS). *Remote Sens. Environ.*, 44(2):127 – 143, 1993.

[67] J. E. Fowler. Compressive pushbroom and whiskbroom sensing for hyperspectral remote-sensing imaging. In *2014 IEEE Int. Conf. Image Process. (ICIP)*, pages 684–688, Oct 2014.

[68] E. Prentice, M. E. Grøtte, F. Sigernes, and T. A. Johansen. Design of hyperspectral imager using COTS optics for small satellite applications. In *Int. Conf. Space Opt. - ICSO 2021*, 2021.

[69] Fred Sigernes, Mikko Syrjäsuo, Rune Storvold, Joao Fortuna, Mariusz Eivind Grøtte, and Tor Arne Johansen. Do it yourself hyperspectral imager for handheld to airborne operations. *Opt. Express*, 26(5):6021–6035, 2018.

[70] D. Jervis, J. McKeever, B. O. A. Durak, J. J. Sloan, D. Gains, D. J. Varon, A. Ramier, M. Strupler, and E. Tarrant. The GHGSat-D imaging spectrometer. *Atmos. Meas. Tech.*, in review, 2020.

[71] S. Henrot, C. Soussen, and D. Brie. Fast Positive Deconvolution of Hyperspectral Images. *IEEE Trans. Image Process.*, 22(2):828–833, 2013.

[72] Susan Kay, John D. Hedley, and Samantha Lavender. Sun Glint Correction of High and Low Spatial Resolution Images of Aquatic Scenes: a Review of Methods for Visible and Near-Infrared Wavelengths. *Remote Sens.*, 1(4):697–730, 2009.

[73] A. G. C. Guerra, F. Francisco, J. Villate, F. Agelet, O. Bertolami, and K. Rajan. On small satellites for oceanography: A survey. *Acta Astronaut.*, 127:404–423, 2016.

[74] M. Lapadatu, S. Bakken, M. E. Grøtte, M. Alver, and T. A. Johansen. Simulation Tool for Hyper-Spectral Imaging From a Satellite. In *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2019.

[75] A. Berk, P. Conforti, R. Kennett, T. Perkins, F. Hawes, and J. van den Bosch. MODTRAN® 6: A major upgrade of the MODTRAN® radiative transfer code. In *2014 6th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2014.

[76] Consultative Committee for Space Data Systems. *Lossless Multispectral and Hyperspectral Image Compression - CCSDS 123.0-B-1*, volume 1. CCSDS: Reston, VA, Blue Book edition, 2012.

[77] Johan Fjeldtvedt and Milica Orlandić. CubeDMA – Optimizing three-dimensional DMA transfers for hyperspectral imaging applications. *Microprocess. Microsyst.*, 65:23–36, March 2019.

[78] Milica Orlandić, Johan Fjeldtvedt, and Tor Arne Johansen. A Parallel FPGA Implementation of the CCSDS-123 Compression Algorithm. *Remote Sensing*, 11(6):673, March 2019.

[79] Johan Fjeldtvedt, Milica Orlandić, and Tor Arne Johansen. An efficient real-time fpga implementation of the ccsds-123 compression standard for hyperspectral images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(10):3841–3852, 2018.

[80] Sivert Bakken, Milica Orlandic, and Tor Arne Johansen. The effect of dimensionality reduction on signature-based target detection for hyperspectral remote sensing. In Thomas S. Pagano, Charles D. Norton, and Sachidananda R. Babu, editors, *CubeSats and SmallSats for Remote Sensing III*, volume 11131, pages 164 – 182. International Society for Optics and Photonics, SPIE, 2019.

[81] M. B. Henriksen, J. L. Garrett, E. F. Prentice, A. Stahl, T. A. Johansen, and F. Sigernes. Real-Time Corrections for a Low-Cost Hyperspectral Instrument. In *2019 10th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5, Amsterdam, Netherlands, September 2019. IEEE.

[82] C. Rodarmel and J. Shan. Principal Component Analysis for Hyperspectral Image Classification. *Surv. Land Inf. Sci.*, 62(2):115–122, 2002.

[83] D. Fernández, C. González, D. Mozos, and S. López. FPGA implementation of the principal component analysis algorithm for dimensionality reduction of hyperspectral images. *J. Real Time Image Process.*, 16:1–12, 2016.

[84] Raffaele Vitale, Anna Zhyrova, João F. Fortuna, Onno E. de Noord, Alberto Ferrer, and Harald Martens. On-The-Fly Processing of continuous high-dimensional data streams. *Chemometrics and Intelligent Laboratory Systems*, 161:118–129, February 2017.

[85] Harald Martens, Jesper Pram Nielsen, and Søren Balling Engelsen. Light Scattering and Light Absorbance Separated by Extended Multiplicative Signal Correction. Application to Near-Infrared Transmission Analysis of Powder Mixtures. *Analytical Chemistry*, 75(3):394–404, February 2003.

[86] D. Manolakis and G. Shaw. Detection algorithms for hyperspectral imaging applications. *IEEE Signal Process. Mag.*, 19(1):29–43, 2002.

[87] Dimitris G. Manolakis. Taxonomy of detection algorithms for hyperspectral imaging applications. *Opt. Eng.*, 44(6):066403–1–11, 2005.

[88] Dordije Boskovic, Milica Orlandić, and Tor Arne Johansen. A reconfigurable multi-mode implementation of hyperspectral target detection algorithms. *Microprocess. Microsyst.*, 78, 2020.

[89] D. Boskovic, M. Orlandić, S. Bakken, and T. A. Johansen. HW/SW Implementation of Hyperspectral Target Detection Algorithm. In *8th Mediterranean Conf. on Embedded Computing – MECO*, pages 1–6, Budva, Montenegro, June 2019. IEEE.

[90] 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 Sens.*, 12(3, 534), 2020.

[91] Y. Zhao, Y. Yuan, and Q. Wang. Fast Spectral Clustering for Unsupervised Hyperspectral Image Classification. *Remote Sens.*, 11(399), 2019.

[92] R. Wang, F. Nie, and W. Yu. Fast Spectral Clustering With Anchor Graph for Large Hyperspectral Images. *IEEE Geosci. Remote Sens. Lett.*, 14(11):2003–2007, 2017.

[93] Mohamed Ismail and Milica Orlandić. Segment-Based Clustering of Hyperspectral Images Using Tree-Based Data Partitioning Structures. *Algorithms*, 13(12, 330), 2020.

[94] Thomas Opsahl, Trym V. Haavardsholm, and Ingebrigt Winjum. Real-time georeferencing for an airborne hyperspectral imaging system. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVII*, volume 8048, pages 290–295. SPIE, 2011.

[95] D. Schläpfer and R. Richter. Geo-atmospheric processing of airborne imaging spectrometry data. Part 1: Parametric orthorectification. *Int. J. Remote Sens.*, 23(13):2609–2630, 2002.

[96] S. C. Park, M. K, Park, and M. G. Kang. Super-resolution image reconstruction: a technical overview. *IEEE Signal Process. Mag.*, 20(3):21–36, 2003.

[97] Joseph L Garrett, Dennis Langer, Karine Avagian, and Annette Stahl. Accuracy of super-resolution for hyperspectral ocean observations. In *OCEANS 2019 - Marseille*, 2019.

[98] Lieven Clarisse, Martin Van Damme, Cathy Clerbaux, and Pierre-François Coheur. Tracking down global NH_3 point sources with wind-adjusted superresolution. *Atmos. Meas. Tech.*, 12:5457–5473, 2019.

[99] Henry Stark and Peyma Oskoui. High-resolution image recovery from image-plane arrays, using convex projections. *Journal of the Optical Society of America A*, 6(11):1715, November 1989.

[100] S. Farsiu, M.D. Robinson, M. Elad, and P. Milanfar. Fast and Robust Multiframe Super Resolution. *IEEE Trans. Image Process.*, 13(10):1327–1344, October 2004.

[101] S. Baker and T. Kanade. Limits on super-resolution and how to break them. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(9):1167–1183, 2002.

[102] T. Köhler, M. Bätz, F. Naderi, A. Kaup, A. Maier, and C. Riess. Toward Bridging the Simulated-to-Real Gap: Benchmarking Super-Resolution on Real Data. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(11):2944–2959, 2019.

[103] J. Yang, J. Wright, T. S. Huang, and Y. Ma. Image Super-Resolution Via Sparse Representation. *IEEE Trans. Image Process.*, 19(11):2861–2873, 2010.

[104] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate Image Super-Resolution Using Very Deep Convolutional Networks. In *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, pages 1646–1654, 2016.

[105] Saeed Anwar, Salman Khan, and Nick Barnes. A Deep Journey into Super-Resolution: A Survey. *ACM Computing Surveys*, 53(3), 2020.

[106] Charis Lanaras, Emmanuel Baltsavias, and Konrad Schindler. Hyperspectral Super-Resolution by Coupled Spectral Unmixing. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 3586–3594, Santiago, Chile, December 2015. IEEE.

[107] N. Yokoya, C. Grohnfeldt, and J. Chanussot. Hyperspectral and Multispectral Data Fusion: A comparative review of the recent literature. *IEEE Geosci. Remote Sens. Mag.*, 5(2):29–56, 2017.

[108] T. Akgun, Y. Altunbasak, and R. M. Mersereau. Super-resolution reconstruction of hyperspectral images. *IEEE Trans. Image Process.*, 14(11):1860–1875, 2005.

[109] Karine Avagian and Milica Orlandić. An Efficient FPGA Implementation of Richardson-Lucy Deconvolution Algorithm for Hyperspectral Images. *Electronics*, 10(4, 504), 2021.

[110] K. Avagian, M. Orlandić, and T. A. Johansen. An FPGA-oriented HW/SW Codesign of Lucy-Richardson Deconvolution Algorithm for Hyperspectral Images. In *8th Mediterranean Conf. on Embedded Computing (MECO)*, 2019.

[111] Howard R. Gordon. Atmospheric correction of ocean color imagery in the Earth Observing System era. *J. Geophys. Res. Atmos.*, 102D(D14):17081–17106, 1997.

[112] Robert S. Fraser, Shana Mattoo, Eueng-Nan Yeh, and C. R. McClain. Algorithm for atmospheric and glint corrections of satellite measurements of ocean pigment. *J. Geophys. Res. Atmos.*, 102:17107–17118, 1997.

[113] Th. Schroeder, I. Behnert, M. Schaale, J. Fischer, and R. Doerffer. Atmospheric correction algorithm for MERIS above case-2 waters. *Int. J. Remote Sens.*, 28(7):1469–1486, 2007.

[114] B. Gao and R. Li. Spectral calibrations ofHICOdata using atmospheric bands and radiance adjustment based onHICOand MODIS data comparisons. In *2010 IEEE Int. Geosci. Remote Sens. Symp.*, pages 4260–4263, 2010.

[115] Knut Stamnes, Børge Hamre, Snorre Stamnes, Nan Chen, Yongzhen Fan, Wei Li, Zhenyi Lin, and Jakob Stamnes. Progress in Forward-Inverse Modeling Based on Radiative Transfer Tools for Coupled Atmosphere-Snow/Ice-Ocean Systems: A Review and Description of the AccuRT Model. *Appl. Sci.*, 8(12, 2682), 2018.

[116] Mariusz E. Grøtte, Roger Birkeland, Evelyn Honoré-Livermore, Sivert Bakken, Joseph L. Garrett, Elizabeth F. Prentice, Fred Sigernes, J. Tommy Gravdal, and Tor A. Johansen. Ocean Color Hyperspectral Remote Sensing with High Resolution and Low Latency – the HYPSO-1 CubeSat Mission. *Submitted*, 2020.

[117] Alberto Dallolio, Bendik Agdal, Artur Zolich, Jo Arve Alfredsen, and Tor Arne Johansen. Long-Endurance Green Energy Autonomous Surface Vehicle Control Architecture. In *OCEANS 2019 MTS/IEEE SEATTLE*, pages 1–10, Seattle, WA, USA, October 2019. IEEE.

[118] Joana Fonseca, Jieqiang Wei, Karl H Johansson, and Tor Arne Johansen. Cooperative Decentralised Circumnavigation with Application to Algalbloom Tracking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[119] Johan Fjeldtvedt, Milica Orlandic, and Tor Arne Johansen. An Efficient Real-Time FPGA Implementation of the CCSDS-123 Compression Standard for Hyperspectral Images. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(10):3841–3852, October 2018.

[120] Karine Avagian, Milica Orlandic, and Tor Arne Johansen. An FPGA-oriented HW/SW Codesign of Lucy-Richardson Deconvolution Algorithm for Hyperspectral Images. *2019*

8th Mediterranean Conference on Embedded Computing (MECO), page 5, 2019.

[121] Joseph L Garrett, Dennis Langer, Karine Avagian, and Annette Stahl. Accuracy of super-resolution for hyperspectral ocean observations. *IEEE OCEANS*, page 8, 2019.

[122] Djorđije Bošković, Milica Orlandić, and Tor Arne Johansen. A reconfigurable multi-mode implementation of hyperspectral target detection algorithms. *Microprocessors and Microsystems*, 78:103258, October 2020.

[123] João Fortuna and Tor Arne Johansen. A lightweight payload for hyperspectral remote sensing using small uavs. In *2018 9th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5, 2018.

[124] Jose Pinto, Paulo S. Dias, Ricardo Martins, Joao Fortuna, Eduardo Marques, and Joao Sousa. The LSTS toolchain for networked vehicle systems. In *2013 MTS/IEEE OCEANS - Bergen*, pages 1–9, Bergen, June 2013. IEEE.

[125] Sigurd M. Albrektsen and Tor Arne Johansen. User-Configurable Timing and Navigation for UAVs. *Sensors*, 18(8):2468, August 2018. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

[126] Gaby Launay. pypyueye, 2017. github.com.

[127] Robert L. Lucke, Michael Corson, Norman R. McGlothlin, Steve D. Butcher, Daniel L. Wood, Daniel R. Korwan, Rong R. Li, Willliam A. Snyder, Curt O. Davis, and Davidson T. Chen. Hyperspectral Imager for the Coastal Ocean: instrument description and first images. *Applied Optics*, 50(11):1501, April 2011.

[128] João Fortuna, Harald Martens, and Tor Arne Johansen. Multivariate image fusion: A pipeline for hyperspectral data enhancement. *Chemometrics and Intelligent Laboratory Systems*, 205:104097, 2020.

[129] Nigel J. H. Smith. *The Amazon River Forest: A Natural History of Plants, Animals, and People*. Number 9780195126839 in OUP Catalogue. Oxford University Press, 1999.

[130] Andy Stock and Ajit Subramaniam. Accuracy of empirical satellite algorithms for mapping phytoplankton diagnostic pigments in the open ocean: A supervised learning perspective. *Frontiers in Marine Science*, 2020.

[131] Heidi Dierssen et. al. Data needs for hyperspectral detection of algal diversity across the globe. *Oceanography*, March 2020.

[132] ZhongPing Lee, Nima Pahlevan, Yu-Hwan Ahn, Steven Greb, and David O'Donnell. Robust approach to directly measuring water-leaving radiance in the field. *Applied Optics*, 52(8):1693–1701, 2013.

[133] Hua Lin, Zhongping Lee, Gong Lin, and Xiaolong Yu. Experimental evaluation of the self-shadow and its correction for on-water measurements of water-leaving radiance. *Appl. Opt.*, 59(17):5325–5334, Jun 2020.

[134] m11keluis/oceanoptics: Kelly's ocean optics toolbox. `https://github.com/m11keluis/oceanoptics`. (Accessed on 12/01/2021).

[135] ZhongPing Lee, Kendall L Carder, and Robert A Arnone. Deriving inherent optical properties from water color: a multiband quasi-analytical algorithm for optically deep waters. *Applied optics*, 41(27):5755–5772, 2002.

[136] Suzanne Roy, Carole A Llewellyn, Einar Skarstad Egeland, and Geir Johnsen. *Phytoplankton pigments: characterization, chemotaxonomy and applications in oceanography*. Cambridge University Press, 2011.

[137] Uv-2401pc/uv-2501pc (shimadzu corporation) | evisa's instruments database. `http://www.speciation.net/Database/Instruments/Shimadzu-Corporation/UV2401PCUV2501PC-;i1300`. (Accessed on 12/02/2021).

[138] P Jeremy Werdell, Bryan A Franz, Sean W Bailey, Gene C Feldman, Emmanuel Boss, Vittorio E Brando, Mark Dowell, Takafumi Hirata, Samantha J Lavender, ZhongPing Lee, et al. Generalized ocean color inversion model for retrieving marine inherent optical properties. *Applied optics*, 52(10):2019–2037, 2013.

[139] John Watson and Oliver Zielinski. *Subsea optics and imaging*. Elsevier, 2013.

[140] S. Dutkiewicz. *Synergy between Ocean Colour and Biogeochemical/Ecosystem Models*, volume No. 19 of *Reports of the International Ocean Colour Coordinating Group*. IOCCG, Dartmouth, Canada, 2020.

[141] Eric Langford. Quartiles in elementary statistics. *Journal of Statistics Education*, 14(3), 2006.

[142] Qaa_v5.pdf. `https://www.ioccg.org/groups/Software_OCA/QAA_v5.pdf`. (Accessed on 02/18/2021).

[143] Kristin Tøndel and Harald Martens. Analyzing complex mathematical model behavior by partial least squares regression-based multivariate metamodeling. *Wiley Interdisciplinary Reviews: Computational Statistics*, 6(6):440–475, 2014.

[144] Guangchun Luo, Guangyi Chen, Ling Tian, Ke Qin, and Shen-En Qian. Minimum noise fraction versus principal component analysis as a preprocessing step for hyperspectral imagery denoising. *Canadian Journal of Remote Sensing*, 42(2):106–116, 2016.

[145] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.

[146] Xiaoying Jin, Scott Paswaters, and Harold Cline. A comparative study of target detection algorithms for hyperspectral imagery. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*, volume 7334, page 73341W. International Society for Optics and Photonics, 2009.

[147] José M Bioucas-Dias and José MP Nascimento. Hyperspectral subspace identification. *IEEE Transactions on Geoscience and Remote Sensing*, 46(8):2435–2445, 2008.

[148] Andrew A Green, Mark Berman, Paul Switzer, and Maurice D Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on geoscience and remote sensing*, 26(1):65–74, 1988.

[149] W Wu, DL Massart, and S De Jong. The kernel pca algorithms for wide data. part i: theory and algorithms. *Chemometrics and Intelligent Laboratory Systems*, 36(2):165–172, 1997.

[150] Frank Westad and Martin Kermit. Cross validation and uncertainty estimates in independent component analysis. *Analytica chimica acta*, 490(1-2):341–354, 2003.

[151] Aapo Hyvärinen, Juha Karhunen, and Erkki Oja. Independent component analysis, adaptive and learning systems for signal processing, communications, and control. *John Wiley & Sons, Inc*, 1:11–14, 2001.

[152] Dimitris Manolakis, Eric Truslow, Michael Pieper, Thomas Cooley, and Michael Brueggeman. Detection algorithms in hyperspectral imaging systems: An overview of practical algorithms. *IEEE Signal Processing Magazine*, 31(1):24–33, January 2014.

[153] José M Bioucas-Dias, Antonio Plaza, Gustavo Camps-Valls, Paul Scheunders, Nasser Nasrabadi, and Jocelyn Chanussot. Hyperspectral remote sensing data analysis and future challenges. *IEEE Geoscience and remote sensing magazine*, 1(2):6–36, June 2013.

[154] Dimitris Manolakis, Ronald Lockwood, Thomas Cooley, and John Jacobson. Is there a best hyperspectral detection algorithm? In *Algorithms and technologies for multispectral, hyperspectral, and ultraspectral imagery XV*, volume 7334, page 733402. International Society for Optics and Photonics, 2009.

[155] Michael E Winter. N-findr: An algorithm for fast autonomous spectral end-member determination in hyperspectral data. In *Imaging Spectrometry V*, volume 3753, pages 266–276. International Society for Optics and Photonics, 1999.

[156] Hyperspectral remote sensing scenes - grupo de inteligencia computacional (gic). `http://www.ehu.eus/ccwintco/index.php/Hyperspectral_Remote_Sensing_Scenes`, 1 2019. (Accessed on 01/11/2019).

[157] Marion F Baumgardner, Larry L Biehl, and David A Landgrebe. 220 band aviris hyperspectral image data set: June 12, 1992 indian pine test site 3. *Purdue University Research Repository*, 10:R7RX991C, 2015.

[158] Robert O Green, Michael L Eastwood, Charles M Sarture, Thomas G Chrien, Mikael Aronsson, Bruce J Chippendale, Jessica A Faust, Betina E Pavri, Christopher J Chovit, Manuel Solis, et al. Imaging spectroscopy and the airborne visible/infrared imaging spectrometer (aviris). *Remote sensing of environment*, 65(3):227–248, 1998.

[159] Andreas A Mueller, Andrea Hausold, and Peter Strobl. Hysens-dais/rosis imaging spectrometers at dlr. In *Remote Sensing for Environmental Monitoring, GIS Applications, and Geology*, volume 4545, pages 225–236. International Society for Optics and Photonics, 2002.

[160] Mark A Folkman, Jay Pearlman, Lushalan B Liao, and Peter J Jarecke. Eo-1/hyperion hyperspectral imager design, development, characterization, and calibration. In *Hyperspectral Remote Sensing of the Land and Atmosphere*, volume 4151, pages 40–52. International Society for Optics and Photonics, 2001.

[161] AM Baldridge, SJ Hook, CI Grove, and G Rivera. The aster spectral library version 2.0. *Remote Sensing of Environment*, 113(4):711–715, 2009.

[162] Spain. Grupo de Inteligencia Computacional, Universidad del País Vasco / Euskal Herriko Unibertsitatea (UPV/EHU). Hyperspectral imagery synthesis (eias) toolbox. `http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Imagery_Synthesis_tools_for_MATLAB`, 2019 07. (Accessed on 07/08/2019).

[163] Dimitris Manolakis, David Marden, Gary A Shaw, et al. Hyperspectral image processing for automatic target detection applications. *Lincoln laboratory journal*, 14(1):79–116, 2003.

[164] D Manolakis, M Pieper, E Truslow, T Cooley, M Brueggeman, and S Lipson. The remarkable success of adaptive cosine estimator in hyperspectral target detection. In Sylvia S. Shen and Paul E. Lewis, editors, *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XIX*, volume 8743, page 874302, Baltimore, Maryland, USA, May 2013. International Society for Optics and Photonics.

[165] Arto Kaarna. Integer pca and wavelet transforms for multispectral image compression. In *IGARSS 2001. Scanning the Present and Resolving the Future. Proceedings. IEEE 2001 International Geoscience and Remote Sensing Symposium (Cat. No. 01CH37217)*, volume 4, pages 1853–1855. IEEE, 2001.

[166] Jannick Kuester, Wolfgang Gross, and Wolfgang Middelmann. An approach to near-lossless hyperspectral data compression using deep autoencoder. In *Image and Signal Processing for Remote Sensing XXVI*, volume 11533, page 1153311. International Society for Optics and Photonics, 2020.

[167] Yaman Dua, Vinod Kumar, and Ravi Shankar Singh. Comprehensive review of hyperspectral image compression algorithms. *Optical Engineering*, 59(9):1 – 39, 2020.

[168] Raúl Guerra, Yubal Barrios, María Díaz, Lucana Santos, Sebastián López, and Roberto Sarmiento. A new algorithm for the on-board compression of hyperspectral images. *Remote Sensing*, 10(3), 2018.

[169] Miguel Hernandez-Cabronero, Aaron Barry Kiely, Matthew Klimesh, Ian Blanes, Jonathan Ligo, Enrico Magli, and Joan Serra-Sagrista. The ccsds 123.0-b-2 low-complexity lossless and near-lossless multispectral and hyperspectral image compression standard: A comprehensive review. *IEEE Geoscience and Remote Sensing Magazine*, pages 0–0, 2021.

[170] Q. Du and J. E. Fowler. Hyperspectral image compression using jpeg2000 and principal component analysis. *IEEE Geoscience and Remote Sensing Letters*, 4(2):201–205, 2007.

[171] Daniel Báscones, Carlos González, and Daniel Mozos. Hyperspectral image compression using vector quantization, pca and jpeg2000. *Remote Sensing*, 10(6):907, 2018.

[172] B. Penna, T. Tillo, E. Magli, and G. Olmo. Transform coding techniques for lossy hyperspectral data compression. *IEEE Transactions on Geoscience and Remote Sensing*, 45(5):1408–1421, 2007.

[173] Ian Blanes and Joan Serra-Sagrista. Cost and scalability improvements to the karhunen–loêve transform for remote-sensing image coding. *IEEE Transactions on Geoscience and Remote Sensing*, 48(7):2854–2863, 2010.

[174] Ian Blanes, Joan Serra-Sagrista, Michael W. Marcellin, and Joan Bartrina-Rapesta. Divide-and-conquer strategies for hyperspectral image processing: A review of their benefits and advantages. *IEEE Signal Processing Magazine*, 29(3):71–81, 2012.

[175] D. Manolakis, R. Lockwood, and T. Cooley. On the spectral correlation structure of hyperspectral imaging data. In *IGARSS 2008 - 2008 IEEE International Geoscience and Remote Sensing Symposium*, volume 2, pages II–581–II–584, 2008.

[176] Daniel Báscones, Carlos González, and Daniel Mozos. An fpga accelerator for real-time lossy compression of hyperspectral images. *Remote Sensing*, 12(16):2563, 2020.

[177] Ernestina Martel, Raquel Lazcano, José López, Daniel Madroñal, Rubén Salvador, Sebastián López, Eduardo Juarez, Raúl Guerra, César Sanz, and Roberto Sarmiento. Implementation of the principal component analysis onto high-performance computer facilities for hyperspectral dimensionality reduction: Results and comparisons. *Remote Sensing*, 10(6), 2018.

[178] P. Hao and Q. Shi. Reversible integer klt for progressive-to-lossless compression of multiple component images. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 1, pages I–633, 2003.

[179] Jing Zhang, James E Fowler, Nicolas H Younan, and Guizhong Liu. Evaluation of jp3d for lossy and lossless compression of hyperspectral imagery. In *2009 IEEE International Geoscience and Remote Sensing Symposium*, volume 4, pages IV–474. IEEE, 2009.

[180] Steven L Brunton and J Nathan Kutz. *Data-driven science and engineering: Machine learning, dynamical systems, and control*. Cambridge University Press, 2019.

[181] Michel Barret, Jean-Louis Gutzwiller, Isidore Paul Akam Bita, and Florio Dalla Vedova. Lossy hyperspectral images coding with exogenous quasi optimal transforms. In *2009 Data Compression Conference*, pages 411–419, 2009.

[182] David R Thompson, Joseph W Boardman, Michael L Eastwood, and Robert O Green. A large airborne survey of earth's visible-infrared spectral dimensionality. *Optics express*, 25(8):9186–9195, 2017.

[183] W. Wang, Z. Zhao, and H. Zhu. Hyperspectral image compression method based on spectral statistical correlation. In *2009 2nd International Congress on Image and Signal Processing*, pages 1–5, 2009.

[184] Yongmin Li. On incremental and robust subspace learning. *Pattern Recognition*, 37(7):1509–1518, 2004.

[185] W. Sun and Q. Du. Hyperspectral band selection: A review. *IEEE Geoscience and Remote Sensing Magazine*, 7(2):118–139, 2019.

[186] Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[187] James e. fowler. `https://my.ece.msstate.edu/faculty/fowler/software.html`. (Accessed on 09/22/2020).

[188] Rafael Gonzalez. *Digital image processing*. Prentice Hall, Upper Saddle River, N.J, 2008.

[189] Marc Antonini, Michel Barlaud, Pierre Mathieu, and Ingrid Daubechies. Image coding using wavelet transform. *IEEE Transactions on image processing*, 1(2):205–220, 1992.

[190] Official repository of the openjpeg project. `https://github.com/uclouvain/openjpeg`. (Accessed on 10/21/2020).

[191] Ntnu-smallsat-lab/hsi-pca-j2k-residuals: A matlab implementation to investigate the effects of adding significant residuals to hsi compression using pca and jpeg2000. `https://github.com/NTNU-SmallSat-Lab/HSI-PCA-J2K-Residuals`.

[192] Robert L Lucke, Michael Corson, Norman R McGlothlin, Steve D Butcher, Daniel L Wood, Daniel R Korwan, Rong R Li, Willliam A Snyder, Curt O Davis, and Davidson T Chen. Hyperspectral imager for the coastal ocean: instrument description and first images. *Applied*

optics, 50(11):1501–1516, 2011.

[193] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. No-reference image quality assessment in the spatial domain. *IEEE Transactions on image processing*, 21(12):4695–4708, 2012.

[194] Aaron B. Kiely and Matthew A. Klimesh. Exploiting calibration-induced artifacts in lossless compression of hyperspectral imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 47(8):2672–2678, 2009.

[195] Jpl | aviris data portal. `https://aviris.jpl.nasa.gov/dataportal/`. (Accessed on 09/15/2020).

[196] Hico data access. `https://oceandata.sci.gsfc.nasa.gov/HICO/L1`. (Accessed on 11/17/2020).

[197] Jun Wang and HK Huang. Three-dimensional image compression with wavelet transforms. In *Handbook of medical imaging*, pages 851–862. Academic Press, Inc., 2000.

[198] Miri Trainic, Ilan Koren, Shlomit Sharoni, Miguel Frada, Lior Segev, Yinon Rudich, and Assaf Vardi. Infection dynamics of a bloom-forming alga and its virus determine airborne coccolith emission from seawater. *iScience*, 6:327–335, 2018.

[199] O. M. Borge. Atmospheric correction over coastal waters based on machine learning models. Master's thesis, NTNU, 2020.

[200] Y. Fan *et al.* Atmospheric correction over coastal waters using multilayer neural networks. *Remote Sensing of Enviroment*, 199:218–240, 2017.

[201] A.D. Gerace *et al.* Increased potential to monitor water quality in the near-shore environment with Landsat's next-generation satellite. *Journal of Applied Remote Sensing*, 7(1):1–19, 2013.

[202] K. Stamnes *et al.* Progress in Forward-Inverse Modeling Based on Radiative Transfer Tools for Coupled Atmosphere-Snow/Ice-Ocean Systems: A Review and Description of the AccuRT Model. *Applied Sciences*, 8:2682, 2018.

[203] K. Moore *et al.* Spectral reflectance of whitecaps: Their contribution to water-leaving radiance. *Journal of Geophysical Research*, 105:6493–6499, 2000.

[204] K. Ruddick. Coastcolour: Round robin protocol, version 1.2. *Brockmann Consult*, 2010.

[205] A. Savitzky and M. J.E. Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[206] Nirpy research. `https://nirpyresearch.com/`. (Accessed on 2020-10-05).

[207] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, 2019.

[208] Roman Rosipal and Nicole Krämer. Overview and recent advances in partial least squares. In *International Statistical and Optimization Perspectives Workshop" Subspace, Latent Structure and Feature Selection"*, pages 34–51. Springer, 2005.

[209] Nasa ocean color. `https://oceancolor.gsfc.nasa.gov/atbd/chlor_a/`. (Accessed on 03-03-2020).

[210] Wesley J. Moses, Anatoly A. Gitelson, Sergey Berdnikov, Jeffrey H. Bowles, Vasiliy Povazhnyi, Vladislav Saprygin, Ellen J. Wagner, and Karen W. Patterson. Hico-based nir–red models for estimating chlorophyll- a concentration in productive coastal waters. *IEEE Geoscience and Remote Sensing Letters*, 11(6):1111–1115, 2014.

[211] Kimberly Ryan and Khalid Ali. Application of a partial least-squares regression model to retrieve chlorophyll-a concentrations in coastal waters using hyper-spectral data. *Ocean Science Journal*, 51(2):209–221, 2016.

[212] Katalin Blix and Torbjørn Eltoft. Machine learning automatic model selection algorithm for oceanic chlorophyll-a content retrieval. *Remote Sensing*, 10(5):775, 2018.

[213] Bernhard Scholkopf and Alexander J Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[214] Hico - sample image gallery. `http://hico.coas.oregonstate.edu/gallery/gallery-scenes.php`. (Accessed on 04-02-2020).

[215] Howard R Gordon. Can the lambert-beer law be applied to the diffuse attenuation coefficient of ocean water? *Limnology and Oceanography*, 34(8):1389–1409, 1989.

[216] John E O'Reilly, Stephane Maritorena, B Greg Mitchell, David A Siegel, Kendall L Carder, Sara A Garver, Mati Kahru, and Charles McClain. Ocean color chlorophyll algorithms for seawifs. *Journal of Geophysical Research: Oceans*, 103(C11):24937–24953, 1998.

[217] Svante Wold, Michael Sjöström, and Lennart Eriksson. PLS-regression: a basic tool of chemometrics. *Chemometrics and intelligent laboratory systems*, 58(2):109–130, 2001.

[218] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[219] M. Törngren and U Sellgren. Complexity challenges in development of cyber-physical systems. In Marjan Sirjani Martin Lohstroh, Patricia Derler, editor, *Principles of Modeling. Lecture Notes in Computer Science*, pages 478–503. Springer, Switzerland, 2018. `https://doi.org/10.1007/978-3-319-95246-8_27`.

[220] Mary A Bone, Mark R Blackburn, Donna H Rhodes, David N Cohen, and Jaime A Guerrero. Transforming systems engineering through digital engineering. *The Journal of Defense Modeling and Simulation*, 16(4):339–355, 2019. https://doi.org/10.1177/2F1548512917751873.

[221] SEBoK. Model-based systems engineering (mbse) (glossary) — sebok,, 2020. URL: "https://www.sebokwiki.org/w/index.php?title=Model-Based_Systems_Engineering_(MBSE)_(glossary)&oldid=59725", Accessed: 2021-02-15.

[222] A. Aigner and A. Khelil. Assessment of model-based methodologies to architect cyber-physical systems. In *International Conference on Omni-Layer Intelligent Systems*, pages 146–151, 2019. https://doi.org/10.1145/3312614.3313779.

[223] K. Beck. Embracing change with extreme programming. *Computer*, 32(10):70–77, 1999. https://doi.org/10.1109/2.796139.

[224] K. Könnöla, S. Suomi, T. Mäkilä, and T. Lehtonen. Can embedded space system development benefit from agile practices? *EURASIP Journal on Embedded Systems*, 1:1–16, 2017. https://doi.org/10.1186/s13639-016-0040-z.

[225] Jøran Grande, Roger Birkeland, Torfinn Lindem, Rune Schlanbusch, Torbjørn Houge, Stian Vik Mathisen, and Kolbjørn Dahle. Educational benefits and challenges for the norwegian student satellite program. In *Proceedings of the 64th International Astronautical Congress*, 2013.

[226] E. Honoré-Livermore and R. Birkeland. Managing product development and integration of a university CubeSat in a locked down world. In *IEEE Aerospace Conference*, Virtual, USA, 2021.

[227] Stefanie Paluch, David Antons, Malte Brettel, Christian Hopp, Torsten-Oliver Salge, Frank Piller, and Daniel Wentzel. Stage-gate and agile development in the digital age: Promises, perils, and boundary conditions. *Journal of Business Research*, 110:495–501, 2020. https://doi.org/10.1016/j.jbusres.2019.01.063.

[228] Robert G. Cooper. The drivers of success in new-product development. *Industrial Marketing Management*, 76:36–47, 2019. https://doi.org/10.1016/j.indmarman.2018.07.005.

[229] Merriam-Webster Dictionary. *Agile*, 2020 (accessed October 13, 2020). https://www.merriam-webster.com/dictionary/agile.

[230] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. *Manifesto for Agile Software*

Development, 2001. URL: https://agilemanifesto.org/, Accessed October 12, 2020.

[231] Greg Wilson, D. A. Aruliah, C. Titus Brown, Neil P. Chue Hong, Matt Davis, Richard T. Guy, Steven H. D. Haddock, Kathryn D. Huff, Ian M. Mitchell, Mark D. Plumbley, Ben Waugh, Ethan P. White, and Paul Wilson. Best practices for scientific computing. *PLOS Biology*, 12(1):e1001745, 2014. https://doi.org/10.1371/journal.pbio.1001745.

[232] Elvira-Maria Arvanitou, Apostolos Ampatzoglou, Alexander Chatzigeorgiou, and Jeffrey C. Carver. Software engineering practices for scientific software development: A systematic mapping study. *Journal of Systems and Software*, 172:110848, 2021.

[233] Magnus Thorstein Sletholt, Jo Hannay, Dietmar Pfahl, Hans Christian Benestad, and Hans Petter Langtangen. A literature review of agile practices and their effects in scientific software development. In *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*, pages 1—-9, Waikiki, Honolulu, HI, USA, 2011. Association for Computing Machinery. https://doi.org/10.1145/1985782.1985784.

[234] Tim Storer. Bridging the chasm: A survey of software engineering practice in scientific programming. *ACM Comput. Surv.*, 50(4), 2017. Article 47, https://doi.org/10.1145/3084225.

[235] Mike Rother and Mark Rosenthal. An approach to becoming agile in a dynamic world. helping employees develop scientific thinking empowers them to solve problems and make decisions. *Association for Manufacturing Excellence: TARGET*, 34, 2018.

[236] Jason H Sharp and Guido Lang. Agile in teaching and learning: Conceptual framework and research agenda. *Journal of Information Systems Education*, 29(2):45–52, 2018. http://jise.org/Volume29/n2/JISEv29n2p45.html.

[237] Zainab Masood, Rashina Hoda, and Kelly Blincoe. Adapting agile practices in university contexts. *Journal of Systems and Software*, 144(Special Issue on Software Engineering Education and Training):501–510, 2018. https://doi.org/10.1016/j.jss.2018.07.011.

[238] K. Lundqvist, A. Ahmed, D. Fridman, and J. Bernard. Interdisciplinary agile teaching. In *2019 IEEE Frontiers in Education Conference (FIE)*, pages 1–8, 2019. https://doi.org/10.1109/FIE43999.2019.9028544.

[239] Mahsood Shah, Leonid Grebennikov, and Chenicheri Sid Nair. A decade of study on employer feedback on the quality of university graduates. *Quality Assurance in Education*, 23:262–278, 2015.

[240] Liv Anne; Støren, Rune Borgan; Reiling, Siv-Elisabeth; Skjelbred, Marte E. S.; Ulvestad, Tone Cecilie; Carlsten, and Dorothy Sutherland Olsen. Utdanning for arbeidslivet: Arbeidsgivers forventninger til og erfaringer med nyutdannede fra universiteter, høgskoler og fagskoler/education for employment: The employers expectations for and experiences with newly graduates from universities, colleges and vocational schools. Technical report, Nordic Institute for Studies in Innovation, Research and Education, 2019.

[241] Sanjay Jayaram and Michael Swartwout. Significance of student-built spacecraft design programs: Its impact on spacecraft engineering education over the last ten years. In *2011 ASEE Annual Conference & Exposition*, 2011.

[242] Elizabeth Howell. Cubesats: Tiny payloads, huge benefits for space research, 2018. https://www.space.com/34324-cubesats.html, Accessed 2020-12-10.

[243] Jeremy Straub and David Whalen. Evaluation of the educational impact of participation time in a small spacecraft development program. *education sciences*, 4(141-154), 2014. https://doi.org/10.3390/educsci4010141.

[244] R. Turner. Toward agile systems engineering processes. *The Journal of Defence Software Engineering*, 20:11–15, 2007.

[245] Tom McDermott, Eileen Van Aken, Nicole Hutchison, Mark Blackburn, Megan Clifford, Neal Chean, Alejandro Salado, and Kaitlin Henderson. Summary report task order wrt-1001: Digital engineering metrics. Technical report, Stevens Institute of Technology, 2020.

[246] N. Garzaniti, S. Briatore, C. Fortin, and A. Golkar. Effectiveness of the scrum methodology for agile development of space hardware. In *2019 IEEE Aerospace Conference*, pages 1–8, 2019. https://doi.org/10.1109/AERO.2019.8741892.

[247] P. M. Huang, A. G. Darrin, and A. A. Knuth. Agile hardware and software system engineering for innovation. In *2012 IEEE Aerospace Conference*, pages 1–10, 2012. https://doi.org/10.1109/AERO.2012.6187425.

[248] Martin Fowler and Matthew Foemmel. Continuous integration, 2006. https://moodle2019-20.ua.es/moodle/pluginfile.php/2228/mod_resource/content/2/martin-fowler-continuous-integration.pdf.

[249] Scott Chacon and Ben Straub. *Pro git*. Springer Nature, 2014. https://doi.org/10.1007/978-1-4842-0076-6.

[250] S. Corpino and F. Stesina. Verification of a cubesat via hardware-in-the-loop simulation. *IEEE Transactions on Aerospace and Electronic Systems*, 50(4):2807–2818, 2014. https://doi.org/10.1109/TAES.2014.130370.

[251] Peter M. B. Waswa and Sangram Redkar. A survey of space mission architecture and system actualisation methodologies. *International Journal of Aerospace Engineering*, 4(3):38, 2017.

https://doi.org/10.1504/IJSPACESE.2017.085674.

[252] Jøran Grande, Roger Birkeland, Amund Gjersvik, and Christoffer Stausland. Norwegian student satellite program - lessons learned. In *Proceedings of The 68th International Astronautical Congress*, 2017.

[253] Dustin Heaton and Jeffrey C. Carver. Claims about the use of software engineering practices in science: A systematic literature review. *Information and Software Technology*, 67:207–219, 2015.

[254] Yang Li. Reengineering a scientific software and lessons learned. In *Proceedings of the 4th International Workshop on Software Engineering for Computational Science and Engineering*, page 41–45, Waikiki, Honolulu, HI, USA, 2011. Association for Computing Machinery.

[255] Karen S. Ackroyd, Steve H. Kinder, Geoff R. Mant, Mike C. Miller, Christine A. Ramsdale, and Paul C. Stephenson. Scientific software development at a research facility. *IEEE Software*, 25(4):44–51, 2008.

[256] Steve McConnell. *Code complete*. Pearson Education, 2004.

[257] Mark W Maier. System and software architecture reconciliation. *Systems Engineering*, 9(2):146–159, 2006.

[258] Carlos E. Gonzalez, Camilo J. Rojas, Alexandre Bergel, and Marcos A. Diaz. An Architecture-Tracking Approach to Evaluate a Modular and Extensible Flight Software for CubeSat Nanosatellites. *IEEE Access*, 7:126409–126429, 2019.

[259] Joar A. Gjersund. A reconfigurable fault-tolerant on-board processing system for the hypso cubesat. Master's thesis, NTNU, 2020.

[260] Magne Hov. Design and implementation of hardware and software interfaces for a hyper-spectral payload in a small. Master's thesis, NTNU, 2019.

[261] Evelyn Honoré-Livermore, Roger Birkeland, Sivert Bakken, Joseph L. Garrett, and Cecilia Haskins. Digital Engineering Management in an Academic CubeSat Project. *Special Issue on Systems Engineering Challenges in Journal of Aerospace Information Systems*, 2021.

[262] International Organization for Standardization. ISO/IEC/IEEE 24748-1:2018(E) Systems and Software engineering life cycle management, 2018.

[263] Semantic versioning 2.0.0 | semantic versioning. https://semver.org/. (Accessed on 07/29/2021).

[264] M.P. Papazoglou. Service-oriented computing: concepts, characteristics and directions. In *Proceedings of the Fourth International Conference on Web Information Systems Engineering, WISE2003*, pages 3–12, 2003.

[265] Hannu Leppinen, Petri Niemelä, Nuno Silva, Henry Sanmark, Henrik Forstén, Adrian Yanes, Rafal Modrzewski, Antti Kestilä, and Jaan Praks. Developing a linux-based nanosatellite on-board computer: Flight results from the aalto-1 mission. *IEEE Aerospace and Electronic Systems Magazine*, 34(1):4–14, 2019.

[266] J. L. Garrett, S. Bakken, E. F. Prentice, D. Langer, F. S. Leira, E. Honoré-Livermore, R. Birkeland, M. E. Grøtte, T. A. Johansen, and M. Orlandić. Hyperspectral Image Processing Pipelines on Multiple Platforms for Coordinated Oceanographic Observation. *11th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, 2021.

[267] Tuva O. Moxnes. A common software framework for a cubesat with multiple payloads. Master's thesis, NTNU, 2021.

[268] J. L. Garrett, S. Bakken, E. F. Prentice, D. Langer, F. S. Leira, E. Honoré-Livermore, R. Birkeland, M. E. Grøtte, T. A. Johansen, and M. Orlandić. Hyperspectral image processing pipelines on multiple platforms for coordinated oceanographic observation. In *2021 11th Workshop on Hyperspectral Imaging and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pages 1–5, 2021.

[269] Carlos L. G. Batista, Tania Basso, Fátima Mattiello-Francisco, and Regina Moraes. Impacts of the space technology evolution in the v v of embedded software-intensive systems. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 1749–1755, 2020.

[270] M Jones, E Gomez, A Mantineo, and UK Mortensen. Introducing ECSS software-engineering standards within ESA. *ESA bulletin*, pages 132–139, 2002.

[271] Cristina del Castillo-Sancho, Gilberto Grassi, Alexander Kinnaird, Aldous Mills, and David Palma. Lessons learned from aiv in esa's fly your satellite! educational cubesat programme. *Proceedings of the AIAA/USU Conference on Small Satellites*, 2021.

[272] Evelyn Honoré-Livermore, Roger Birkeland, Sivert Bakken, Joseph L Garrett, and Cecilia Haskins. Digital engineering development in an academic cubesat project. *Journal of Aerospace Information Systems*, pages 1–12, 2021.

[273] Jonis Kiesbye, David Messmann, Maximilian Preisinger, Gonzalo Reina, Daniel Nagy, Florian Schummer, Martin Mostad, Tejas Kale, and Martin Langer. Hardware-In-The-Loop and Software-In-The-Loop Testing of the MOVE-II CubeSat. *Aerospace*, 6(12):130, December 2019.

[274] Check | unit testing framework for c. `https://libcheck.github.io/check/`. (Accessed on 07/28/2021).

[275] Arden Albee, Steven Battel, Richard Brace, Garry Burdick, John Casani, Jeffrey Lavell, Charles Leising, Duncan MacPherson, Peter Burr, and Duane Dipprey. Report on the Loss of the Mars Polar Lander and Deep Space 2 Missions. NASA STI/Recon Technical Report N, March 2000.

[276] Maridalsvannet | skiforeningen. https://www.skiforeningen.no/utimarka/omrader/nordmarka-syd/steder/maridalsvannet/. (Accessed on 12/08/2021).

[277] Gara Quintana-Diaz, Roger Birkeland, Torbjörn Ekman, and Evelyn Honoré-Livermore. An SDR Mission Measuring UHF Signal Propagation and Interference between Small Satellites in LEO and Arctic Sensors. In *Small Satellite Conference*, Logan, UT, 2019.

[278] N. Garzaniti and A. Golkar. Performance assessment of agile hardware co-development process. In *2020 IEEE International Symposium on Systems Engineering (ISSE)*, pages 1–6, 2020. https://doi.org/10.1109/ISSE49799.2020.9272209.

[279] Y. M. Li, C. j. Li, Ran Zheng, Xiao Li, and Jun Yang. The research on image processing technology of the star tracker. In *International Symposium on Optoelectronic Technology and Application 2014: Image Processing and Pattern Recognition*, volume 9301, pages 9–13. SPIE, 2014.

[280] T. S. Rose, D. W. Rowen, S. LaLumondiere, N. I. Werner, R. Linares, A. Faler, J. Wicker, C. M. Coffman, G. A. Maul, D. H. Chien, A. Utter, R. P. Welle, and S. W. Janson. Optical communications downlink from a 1.5U Cubesat: OCSD program. In Zoran Sodnik, Nikos Karafolas, and Bruno Cugny, editors, *International Conference on Space Optics — ICSO 2018*, volume 11180, pages 201 – 212. International Society for Optics and Photonics, SPIE, 2019.

[281] Ayman Habib, Youkyung Han, Weifeng Xiong, Fangning He, Zhou Zhang, and Melba Crawford. Automated ortho-rectification of UAV-based hyperspectral data over an agricultural field using frame RGB imagery. *Remote Sens.*, 8(10, 796), 2016.

ISBN 978-82-326-5963-0 (printed ver.)
ISBN 978-82-326-5356-0 (electronic ver.)
ISSN 1503-8181 (printed ver.)
ISSN 2703-8084 (online ver.)

NTNU
Norwegian University of
Science and Technology