A MODULAR HYPERSPECTRAL IMAGE PROCESSING PIPELINE FOR CUBESATS

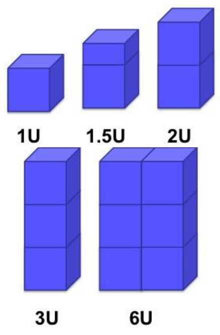
Sivert Bakken[1,4], Aksel Danielsen[1], Kristine Døsvik[2], Joseph L. Garrett[1],
Milica Orlandic[2], Dennis Langer[3], and Tor Arne Johansen [1]

Center for Autonomous Marine Operations and Systems
Department of Engineering Cybernetics(1),
Department of Electronic Systems(2),
Department of Marine Technology(3),
Norwegian University of Science and Technology (NTNU),
and SINTEF OCEAN (4)

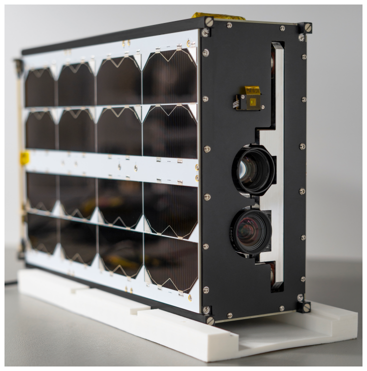
ABSTRACT

This work describes the onboard image processing pipeline that is designed to be used onboard a CubeSat with a Hyper Spectral (Imager/Image) (HSI), namely the HYPSO-1 satellite. While the system itself supports both the use of a Field Programmable Gate Array (FPGA) and CPUs for processing, this work focuses on the latter. A reference to the source code is also enclosed within the text. Some of the planned and developed modules are presented, along with the design choices needed to accommodate the computationally constrained system. The execution time of the pipeline on target hardware with representative hyperspectral data is also shown. The results indicate that the modular pipeline framework is suitable for deployment onboard HYPSO-1.

Index Terms— Hyperspectral, Image, Processing, Edge-Computing



(a)



(b)

Fig. 1: CubeSats; 1a shows different form factors for CubeSats, while 1b shows the actual flight model version of HYPSO-1 prior to launch, which uses a 6U configuration.

1. INTRODUCTION

CubeSats are small satellites that aim to provide easy access to space. They follow the CubeSat design standard created by California Polytechnic State University and Standford University. The standardization of CubeSats enables the use of mass-produced and commercial-of-the-shelf components, which reduces cost. The standard also reduces launch costs compared to custom-built satellites [1]. The CubeSat standard defines the dimensions to be multiples of 10-by-10-by-10 cm units, see 1.

The HYPer-spectral SmallSat for Ocean observation (HYPSO) project aims to enable rapid and continuous capturing and monitoring of algae blooms along the Norwegian coast by launching a 6 unit CubeSat equipped with a HSI [2]. The HYPSO-1 satellite was successfully launched on the 13. of January 2022. HYPSO-1 is the first of several planned satellites for ocean imaging in the HYPSO project. In Fig. 1 an illustration of different CubeSat sizes are given alongside a photo of the Flight Model of HYPSO-1. The mission operation plan aims for the satellite to stay operational for five years. Currently, it orbits the earth and down-links raw and processed hyperspectral data for further analysis. In-depth analysis of initial in-orbit results and first images is a subject for future publications. This work uses the target hardware of the the satellite to estimate the computation time.

A central part of the HYPSO-1 concept is the onboard processing pipeline. A typical hyperspectral datacube can have a large data volume relative to the available downlink bandwidth, which mainly relies on S-band radio communication. For HYPSO-1, the standard data cube after binning is $(l, f, b) = (956, 684, 120)$, i.e., lines, frames and bands. Due to constraints in power usage and communication it is

This work is supported by the Norwegian Research Council (grant no. 270959), the Centre of Autonomous Marine Operations and Systems (NTNU AMOS, grant no.223254), the Norwegian Space Center, the European Space Agency (PRODEX - 4000132515), and NO Grants 2014 – 2021, under project ELO-Hyp, contract no. 24/2020.
Corresponding author: `sivert.bakken@ntnu.no`

desirable to do data processing onboard to reduce the size, increase information throughput and enhance the data for further analysis on the ground [2]. In order to facilitate flexibility in the onboard image processing pipeline executable, this work describes how it as been designed to allow configuration and toggling of the different pipeline modules while in space. This modularity will allow for choosing between different pipelines for data processing based on specific use cases for the end-user and operational resources available. The operational resources include time, memory, and power currently available onboard the satellite. The software architecture design, made to support this re-configuration is described in [3]. This is a partial implementation of the concept discussed in [4].

2. BACKGROUND

Satellites equipped with HSIs are no longer novel in and of themselves. There have been several successful missions in the past, including the EO-1 Hyperion launched by NASA [5] and the HyperScout imager onboard ESA's GomX-4B CubeSat [6].

In [7] a HSI is equipped onboard an aircraft to detect methane gas. They process large amounts of data and detection in real-time by exploiting parallel detection on spatially independent pixels in the pipeline. Their processing pipeline is fixed to a single task, and the available processing power on an aircraft is significantly higher than on a satellite. The pipeline on board HYPSO-1 requires a more flexible implementation and has limited processing capacity. The bands of the cube are processed in parallel in a modular pipeline design. The approach of HyperScout is similar to the design described here [6]. However, as the HYPSO project wants to utilize processing that exploits spectral patterns in spatial pixels, it is inherently inefficient and unwieldy to do processing band by band. In [8] the synergies between compression and anomaly detection are demonstrated and combined in a data pipeline to increase performance. This exciting concept could also be included in the pipeline proposed here.

With the proposed modular hyperspectral image processing pipeline for CubeSats presented here, the goal is to increase the information throughput. However, with the calculated performance of the imager onboard HYPSO-1, given in [9], it is anticipated that the novel mission design can provide end-users of ocean color data with new and novel data products, especially when used in a concert with other autonomous water sampling agents as discussed in [2, 10].

The increased information throughput can come in many forms. Initially, the concept of serving complete data products computed onboard the satellite was discussed in [2]. From the experiences of operating the satellite, it is evident that intermediate data products, e.g., a panchromatic or regular RGB image, can also be beneficial to get an early indication of the image quality.

3. IMPLEMENTATION

The On-board Processing Unit (OPU) is a PicoZed 7030 System on Module, which has 1 GB of DDR3 SDRAM and is based on the Zync-7000 System-on-Chip (SoC) provided by AVNET [2]. The SoC features a Dual-core ARM Cortex A9 running at 667 MHz with 32 KB Level 1 cache for each Central Processing Unit (CPU), as well as a shared 512 KB Level 2 cache. Both cores has the NEON architecture extension for Single instruction, multiple data (SIMD) instructions [3]. Utilizing the NEON engines enables vectorizing data operations, significantly accelerating the image processing algorithms. The software is developed to be run on ARM architecture; it is portable to other systems using ARM architecture for hyperspectral image processing. The code can be made available upon request [11].

This section describes the different processing modules that are developed for HYPSO-1. An overview of the modules can be seen in Tab. 1. More details can be found in [12]. Programmatically, the modules have a similar structure and execution. It starts with parsing the module configuration file, then allocates the memory needed by the data structures and populates them with the model binary data from the disk as specified by the configuration file. Then the given algorithm is applied to the input cube, and subsequent results are stored on an SD card, and the memory is freed.

The metric Spectrograms per Second (SpS) for the modules in Tab. 1 that are dependent on module configuration is given as *N/A*.

Tab. 1: Overview of the different modules, their target implementation and the maximum time of processed SpS during nominal operations for the test setup, rounded down.

Module	Target	SpS
Lossless Compression	CPU/FPGA	4780
Smile and Keystone Correction	CPU	105
RGB Render	CPU	373
Principal Components	CPU	N/A
Classification	CPU	N/A
Clustering	CPU	N/A

3.1. Lossless Compression

An implementation of the CCSDS123.1 standard, both for CPU and FPGA is available on the HYPSO-1 satellite, and has been since launch. A thorough description of the FPGA implementation is given in [13]. This lossless compression algorithm has been used to downlink data cubes of various sizes more efficiently. As Indicated in [13], the algorithm is not well suited to be run on CPUs, but smaller cube sizes have still been compressed without pushing the limits of the HYPSO-1 power budget. The SpS of this module is reported in Tab. 1.

3.2. Smile and Keystone Correction

HSI cameras built from Commercial-Off-The-Shelf (COTS) parts often suffer from optical distortions [14]. These distortions are specific to the camera. The effects warp the image along both the spatial and spectral axis. The spectral effect is called "smile", while the spatial effect is called "keystone". It is possible to correct these distortions in software post-capture. This correction can improve subsequent processing tasks.

Here, the correction is done by enumerating all the pixel indices and calculating the intensity of a specific pixel by finding the intensity of the corresponding pixel in the raw spectrogram. In [14], the intensity of the corresponding pixel is found by estimating a mapping function $f_{SnK} : R^2 \to R^2$ of the pixels between the raw and the desired spectrogram. This function f_{SnK} gives the indices (y_r, λ_r) of the pixel in the raw spectrogram as a function of the indices in the corrected spectrogram (y, λ):

$$(y_r, \lambda_r) = f(y, \lambda). \tag{1}$$

Here, the mapping f_{SnK} is approximated using a second-degree polynomial. The values refer to an image coordinate in between captured pixel values. Bi-linear interpolation gives the final value of the pixel in the corrected image.

The implementation needs to accommodate the requirements for CPU and power usage of the system [2]. The data is stored in Band Interleaved by Pixel (BIP) format, which is ideal for the Smile and Keystone correction since all the data for a single frame is sequential in memory. Pre-computing the weights relevant for the HYPSO-1 spectrograms, used by the bi-linear interpolation, an adaption form the approach found in [14], leads to a significant performance improvement in terms of execution time.

3.3. RGB Render

Experiences from early operations quickly indicated that it is useful to be able to generate a true or false color composite images from the captured image cubes. This module does that and enables the user to configure spatial binning, center wavelengths for the primary colors, and their bandwidth in terms of surrounding bands to be included.

3.4. Principal Component Transform

The configuration file of the dimensionality reduction module specifies the number of principal components d to be returned by the projection using a transformation matrix data. Commonly principal component transform first centers the data and then calculate the projection. Observe that

$$\begin{aligned}\mathbf{c} = \mathbf{P}\widetilde{\mathbf{x}} &= \mathbf{P}(\mathbf{x} - \overline{\mathbf{x}}) \\ &= \mathbf{P}\mathbf{x} - \mathbf{P}\overline{\mathbf{x}} \\ &= \mathbf{P}\mathbf{x} - \overline{\mathbf{c}}.\end{aligned} \tag{2}$$

This operation is computationally cheaper as the projection subspace of $\mathbf{c}$ is lower than the dimension of the spectral pixel vector $\mathbf{x}$. Therefore, the binary file needs to contain the projected mean $\overline{\mathbf{c}}$ and the $d \times b$ projection matrix $\mathbf{P}$. The first $4 \times m$ bytes of the binary file is $\overline{\mathbf{c}}$ and the remaining $4 \times d \times b$ bytes is $\mathbf{P}$. By using a transformation matrix that is computed on ground to reduce the cube dimensionality, it is possible to reduce the execution time of subsequent algorithms as well. Here the dimensionality reduction is used in together with classification to demonstrate this, see [12, 15], and results in Fig. 2.

3.5. Classification

The module is executed by first evaluating the kernel K with the new pixel and the support vectors $\mathbf{S}$. The results are stored as an array where the i-th entry in the array corresponds to $K(\mathbf{x}, \mathbf{s}_i)$. Pre-computing the kernel halves the execution time needed as the kernel computation can be reused when a support vector is used in more than one classifier. As the support vectors are sorted on class, the kernel values for the support vectors of class k start at the offset $\sum_{i=1}^{k} n_{si}$ from the start at fathe array, where n_{si} is the number of support vectors for class i.

The configuration file includes a definition of the trained classifiers, the support vectors $\mathbf{S}$ and coefficients $\boldsymbol{\alpha}\mathbf{y}$ for the classifiers. The first $4 \times d \times n_s$ bytes are the support vectors, followed by the coefficients for the 1v1 classifiers. With $k(k-1)/2$ classifier comparisons, each requiring coefficients for each support vector for a total of $4 \times k(k-1)/2 \times n_s$ bytes. The number of classes, parameters for the kernel function, and the number of support vectors are also defined in the configuration file.

3.6. Clustering

For each spectral pixel, the module enumerates all the nodes and computes the distance to them. If this distance is lower than the current lowest distance, it is set as the lowest distance, and the current label estimate is updated. After all of the nodes have been checked, the label is assigned.

The module configuration file contains the Self-Organizing Map (SOM) nodes and labels. The first $4 \times d \times z^2$ bytes, with d as dimension and z as clusters, are the floating-point node values, while the remaining data are labels. Each label is only stored as one unsigned byte. The number of unique clusters are not expected to exceed 256 for any relevant application. The size of the SOM grid is given by z^2.

3.7. Planned Modules

The modular framework of both the HYPSO-1 satellite and the On-board Image Processing (OBIP) pipeline discussed here allows for further expansion and exploration. There are modules under development, e.g., sub-sampling of cubes to support compressive sensing, rudimentary atmospheric correction to retrieve approximations of measured reflectance for further processing, and more dimensionality reduction approaches for flexible lossy compression, and more. The framework allows the modules to be developed decoupled from other parts while determining their execution order from the configuration files.

4. RESULTS

This section presents the results of the implemented pipelines with their design considerations and their impact. This paper focuses on the execution time of the different modules, not the performance of the algorithms themselves.

4.1. Performance of Smile and Keystone Correction

In terms of performance, reducing computation time by pre-computing the weights reduced the computation time by a factor of 20. This design is a space-time trade-off as storing weights will increase memory usage. The datatypes of the index and each of the weights take 4 bytes, so it totals 20 bytes per pixel. The max supported spectrogram resolution of $(l, b) = (1216, 1936)$ gives a memory overhead of $\approx$ 44MiB. For a nominal capture resolution of $(l, b) = (684, 120)$ is less than 1.57 MiB. Since the data is binned before processing on the satellite, this reduces memory overhead directly proportional to the binning factor. From profiling the total memory usage will be $\leq$ 10MB, which is very feasible for the OPU [12]. The execution time in terms of processed SpS is given in Tab. 1.

4.2. Performance of RGB rendering.

The module can render color composite images from the captured cubes, both as raw images and after smile and keystone correction. The resulting image is encoded to .png format. A bandwidth of three bands was used with the performance indicated in Tab. 1.

4.3. Performance of Principal Component Transform

The computational time of projecting a cube scales linearly with the number of projected components d, as seen in Fig. 2a. On the target hardware with the nominal cube dimensions, the resulting expected computational time in seconds can be given as $E_{pca} = 0.8858d$.

4.4. Performance of Classification

The SVM with the RBF kernel has desirable performance, is relatively fast, and has low memory usage [12]. With $S_{sv} = 105$ support vectors the memory overhead is approximately $4 \times S_{sv} \times \lambda \approx$ 4.8MB for the standard cube with $\lambda = 120$. Moreover, the use of the kernel provides an easy way to modify and experiment with the algorithm without significant modification due to the rest of the algorithm not changing. With d as the number of bands and n_s as the number of support vectors the Execution time can be approximated as $E_{svm} = 0.0235dn_s$ from the experimental data given in Fig. 2b.

4.5. Performance of clustering

Clustering by the use of SOM is approximated to have a quadratic run-time in the dimension z, i.e. the number of clusters, and also scales with the number of bands d. The execution time can be expressed in seconds as $E_{SOM} = 0.0104dz^2$, from the experimental data given in Fig. 2c With $d = 120$, the upper limit of $z \approx 11$, while taking $d = 10$ allows for $z \approx 40$.

5. DISCUSSION & CONCLUSION

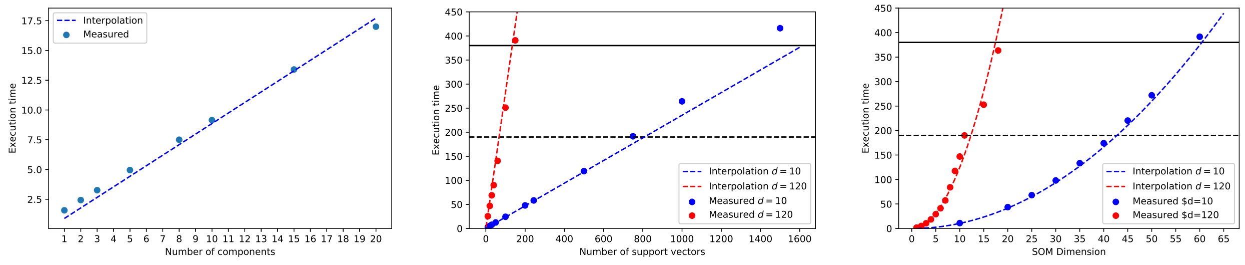
The memory and computational overhead introduced by the pipeline framework are adapted to the target hardware. It keeps the hyperspectral cube in RAM and passes it by reference between modules to avoid expensive copies. The data structures are lightweight and use pre-compute values whenever suitable. By keeping the pipeline as a separate process, its development will not interfere with the software to operate the satellite. The execution times presented here are given under regular operational load.

Future satellite software updates will include the pipeline. The impact of this extension is expected to be beneficial for information throughput and provide insight into how it can be improved further.

6. REFERENCES

[1] CubeSat NASA, "101: Basic concepts and processes for first-time cubesat developers," *NASA and California Polytechnic State University*, 2017.

[2] M. E. Grøtte, R. Birkeland, E. Honoré-Livermore, S. Bakken, J. L. Garrett, E. F. Prentice, F. Sigernes, M. Orlandić, J. T. Gravdahl, and T. 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*, vol. 60, pp. 1–19, 2022.



(a) Principal Component Transform

(b) Support Vector Machine (SVM) Classification.

(c) SOM Clustering.

Fig. 2: Execution time trade-off. Cube dimensions are $(l, f, b) = (684, 956, 120)$. Solid and dashed lines are maximum and desired execution time of HYPSO-1, respectively, given the power budget. Furthere detials can be found in [12, 15]

[3] S. Bakken, E. Honoré-Livermore, R. Birkeland, M. Orlandić, E. F Prentice, J. L Garrett, D. D. Langer, C. Haskins, and T. A. Johansen, "Software development and integration of a hyperspectral imaging payload for hypso-1," in *2022 IEEE/SICE International Symposium on System Integration (SII)*. IEEE, 2022, pp. 183–189.

[4] 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)*, 2021, pp. 1–5.

[5] E. Middleton, S. Ungar, D. Mandl, L. Ong, S. Frye, P. Campbell, D. Landis, J. Young, and N. Pollack, "The earth observing one (eo-1) satellite mission: Over a decade in space," *IEEE JOURNAL OF SELECTED TOPICS IN APPLIED EARTH OBSERVATIONS AND REMOTE SENSING*, vol. 6, pp. 243, 04 2013.

[6] M. et al. Soukup, "Hyperscout: Onboard processing of hyperspectral imaging data on a nanosatellite," in *Small Satellites, System & Services Symposium (4S) 2016*, 05 2016.

[7] D. R. Thompson, I. Leifer, H. Bovensmann, M. Eastwood, M. Fladeland, C. Frankenberg, K. Gerilowski, R. O. Green, S. Kratwurst, T. Krings, B. Luna, and A. K. Thorpe, "Real-time remote detection and measurement for airborne imaging spectroscopy: a case study with methane," *Atmospheric Measurement Techniques*, vol. 8, no. 10, pp. 4383–4397, 2015.

[8] M. Díaz, R. Guerra, P. Horstrand, S. López, J. F. López, and R. Sarmiento, "Towards the concurrent execution of multiple hyperspectral imaging applications by means of computationally simple operations," *Remote Sensing*, vol. 12, no. 8, 2020.

[9] E. F. Prentice, M. E. Grøtte, F. Sigernes, and T. A. Johansen, "Design of a hyperspectral imager using COTS optics for small satellite applications," in *International Conference on Space Optics — ICSO 2020*, Bruno Cugny, Zoran Sodnik, and Nikos Karafolas, Eds. International Society for Optics and Photonics, 2021, vol. 11852, pp. 2154 – 2171, SPIE.

[10] A. Dallolio, G. Quintana-Diaz, E. Honoré-Livermore, J. L. Garrett, R. Birkeland, and T. A. Johansen, "A satellite-usv system for persistent observation of mesoscale oceanographic phenomena," *Remote Sensing*, vol. 13, no. 16, 2021.

[11] NTNU SmallSat Lab, "Modular hyperspectral image processing pipeline for hypso-1," `https://github.com/NTNU-SmallSat-Lab/onboard-pipeline-modules/`, (Accessed on 03/24/2022).

[12] A. Danielsen, "Clustering and classification of hyperspectral images on the hypso cubesat," M.S. thesis, NTNU, 2021.

[13] J. Fjeldtvedt, M. Orlandić, and T. A. 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*, vol. 11, no. 10, pp. 3841–3852, 2018.

[14] 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)*, 2019, pp. 1–5.

[15] A. S. Danielsen, T. A. Johansen, and J. L. Garrett, "Self-organizing maps for clustering hyperspectral images on-board a cubesat," *Remote Sensing*, vol. 13, no. 20, 2021.