JYU DISSERTATIONS 168

Matti Eskelinen

Computational Methods for Hyperspectral Imaging Using Fabry–Perot Interferometers and Colour Cameras

UNIVERSITY OF JYVÄSKYLÄ
FACULTY OF INFORMATION TECHNOLOGY

JYU DISSERTATIONS 168

Matti Eskelinen

Computational Methods for Hyperspectral Imaging Using Fabry–Perot Interferometers and Colour Cameras

Esitetään Jyväskylän yliopiston informaatioteknologian tiedekunnan suostumuksella julkisesti tarkastettavaksi yliopiston Agora-rakennuksen auditoriossa 2 joulukuun 12. päivänä 2019 kello 12.

Academic dissertation to be publicly discussed, by permission of the Faculty of Information Technology of the University of Jyväskylä, in building Agora, auditorium 2, on December 12, 2019 at 12 o'clock noon.

JYVÄSKYLÄN YLIOPISTO
UNIVERSITY OF JYVÄSKYLÄ

JYVÄSKYLÄ 2019

Editors
Timo Männikkö
Faculty of Information Technology, University of Jyväskylä
Ville Korkiakangas
Open Science Centre, University of Jyväskylä

Permanent link to this publication: http://urn.fi/URN:ISBN:978-951-39-7967-6

ISBN 978-951-39-7967-6 (PDF)
URN:ISBN:978-951-39-7967-6
ISSN 2489-9003

ABSTRACT

Eskelinen, Matti
Computational methods for hyperspectral imaging using Fabry–Perot interferometers and colour cameras
Jyväskylä: University of Jyväskylä, 2019, 44 p. (+included articles)
(JYU Dissertations
ISSN 2489-9003; 168)
ISBN 978-951-39-7967-6 (PDF)

Recent research into new technologies for hyperspectral imaging has produced small imagers capable of very fast capture of spectral and spatial information. A design based on an electronically tunable Fabry–Perot interferometer combined with existing camera technology has been developed by VTT and is being utilized in novel applications, such as drone based and handheld hyperspectral imaging. The design allows very fast capture of hyperspectral image cubes with great spatial resolution using either a monochromatic or a colour filter array image sensor. The latter allows imaging speed and wavelength range to be further extended by computing multiple narrowband images from a single exposure.

This research describes the process of computing spectroscopic data using these types of imagers and introduces software tools developed by the author for this purpose. The included articles present solutions developed during the research for building analysis software for hyperspectral imaging using high level languages. They also document computational challenges that need to be considered when utilizing colour filter arrays for hyperspectral imaging and demonstrate the feasibility of this type of imager for use in drone based imaging and laboratory conditions. The software libraries produced during the research are made publicly available under free licenses to facilitate development of new hyperspectral imaging applications using this technology.

Keywords: Hyperspectral imaging, colour filter array, Fabry–Perot interferometer, software development, data analysis, machine learning

TIIVISTELMÄ (ABSTRACT IN FINNISH)

Eskelinen, Matti
Laskennallisia menetelmiä hyperspektrikuvantamiseen Fabry–Perot -interferometrien ja värikameroiden avulla
Jyväskylä: University of Jyväskylä, 2019, 44 s. (+artikkelit)
(JYU Dissertations
ISSN 2489-9003; 168)
ISBN 978-951-39-7967-6 (PDF)

Viime aikoina uusiin hyperspektrikuvantamisen teknologioihin kohdistunut tutkimus on tuottanut nopeita ja pieniä spektrikameroita. Suomessa VTT on kehittänyt sähköisesti ohjattavia Fabry–Perot -interferometrejä ja olemassaolevia kamerateknologioita yhdistäviä spektrikameroita, joita on sovellettu uusissa lennokkipohjaisissa ja käsivaraisissa spektrikuvantamisalustoissa. Tekniikalla voidaan kuvata spektrikuutioita nopeasti käyttämällä joko monokromaattisia tai värisuodattimilla varustettuja kamerakennoja. Jälkimmäisten avulla useita aallonpituuskaistoja voidaan kuvata yhdellä valotuksella, mikä lisää kuvantamisnopeutta ja kasvattaa kuvattavissa olevaa aallonpituusaluetta.

Tämä teos käy läpi tekijän tähän tarkoitukseen kehittämät menetelmät ja ohjelmistotyökalut. Liitetyissä artikkeleissa esitellään tutkimuksen aikana kehitettyjä ratkaisuja spektridataa analysoivien ohjelmistojen rakentamiseen käyttäen korkean tason ohjelmointikieliä. Tämän lisäksi artikkeleissa käydään läpi laskennallisia haasteita, joita värisuodattimilla varustettujen kennojen käyttö tässä yhteydessä aiheuttaa, ja osoitetaan niillä toteutettujen kameroiden soveltuvuus lennokki- ja laboratoriosovelluksissa. Tutkimuksessa tuotettujen ohjelmistojen avoimella julkaisulla pyritään edesauttamaan teknologian käyttöä tulevissa hyperspektrikuvantamisen sovelluksissa.

Avainsanat: Hyperspektrikuvantaminen, värisuodatinmatriisi, Fabry–Perot interferometri, ohjelmistokehitys, data-analyysi, koneoppiminen

Author	Matti Eskelinen Faculty of Information Technology University of Jyväskylä Finland `matti.a.eskelinen@gmail.com`
Supervisors	Professor Pekka Neittaanmäki Faculty of Information Technology University of Jyväskylä Finland
	Docent Ilkka Pölönen Faculty of Information Technology University of Jyväskylä Finland
Reviewers	Professor Markku Hauta-Kasari School of Computing University of Eastern Finland Finland
	Associate Professor Konstantinos Karantzalos Remote Sensing Laboratory National Technical University of Athens Greece
Opponent	Professor Emeritus Jussi Parkkinen Institute of Photonics University of Eastern Finland Finland

ACKNOWLEDGEMENTS

The journey that has led to me finishing this dissertation has been one of opportunities taken, directions changed and setbacks that have lead to new discoveries, and I am deeply grateful to my supervisors Ilkka Pölönen and Pekka Neittaanmäki for all of their support along the way. I must thank Ilkka for first taking me in as a research assistant while I was still finishing my master's, and for guiding me on my path as a researcher ever since. I also thank him for his diligence in securing funding for me and his other henchmen and allowing us to pursue our research to the best of our abilities without having to worry of the more mundane matters of research work. On that note, I extend my gratitude to the Academy of Finland, Jane and Aatos Erkko Foundation, and Business Finland for the funding that has enabled my research. I wish to thank Pekka for persuading me to start my doctoral studies, and for his leadership in pushing me to complete this dissertation and the other mundanities required from doctoral students. Without you, this work would surely still be in the making, if started at all. I must also thank associate professor Konstantinos Karantzalos and professor Markku Hauta-Kasari for their reviews of this dissertation, and professor emeritus Jussi Parkkinen for accepting the invitation to be my opponent.

I wish to thank my past and present colleagues at the laboratory for all the on- and off-topic discussions we've had at the office, and those of you who have accompanied me for the adventures we've had abroad. Thank you Leevi Annala, Billy Braithwaite, Anna-Maria Hakola, Jyri Hämäläinen, Severi Jääskeläinen, Sampsa Kiiskinen, Samuli Rahkonen, Kimmo Riihiaho and Jaana Räisänen for making the hours of writing and coding much less lonely. So long, and thanks for all the catfish. I also thank David Allen for accepting my proposal to spend three months at NIST Gaithersburg and so graciously hosting me there, and him, Clarence, John and other colleagues for the many insights I gained from our discussions and work there. I also thank the people of the GRA and all the lovely people I met there for the company in and outside the campus and the many adventures we endured together. You made my stay in the US the most memorable months of my life so far, and I hope there are still many more memories to be made when we meet again.

My parents deserve thanks not only for my upbringing that has lead me to be a researcher, but for providing me a place in the countryside to retreat to whenever I've wanted to escape the office, the city or my studies. I also thank all my friends who have been there for me throughout my years of study, many of you who I haven't been meeting as often as I would've liked to. Especially Jari, who I wish could still be here. I would much rather be reading you acknowledging all the support that allowed you to see it through, but it didn't. We all miss you.

On a train in Norway, November 2019
Matti Eskelinen

ACRONYMS

ADU	Analog-to-Digital Unit
CCD	Charge Coupled Device
CMOS	Complementary Metal-Oxide-Semiconductor
DN	Digital Number
ESA	European Space Agency
FPI	Fabry–Perot Interferometer
JPL	Jet Propulsion Laboratory
MEMS	MicroElectroMechanical System
SVM	Support Vector Machine
SWIR	Short-wave Infrared
VNIR	Visual and Near Infrared
VTT	VTT Technical research centre of Finland

LIST OF FIGURES

LIST OF TABLES

CONTENTS

LIST OF INCLUDED ARTICLES

PI Eija Honkavaara, Matti Eskelinen, Ilkka Pölönen, Heikki Saari, Harri Ojanen, Rami Mannila, Christer Holmlund, Teemu Hakala, Paula Litkey, Tomi Rosnell, Niko Viljanen, Merja Pulkkanen. Remote Sensing of 3-D Geometry and Surface Moisture of a Peat Production Area Using Hyperspectral Frame Cameras in Visible to Short-Wave Infrared Spectral Ranges Onboard a Small Unmanned Airborne Vehicle (UAV). *IEEE Transactions on Geoscience and Remote Sensing, 54 (9).* DOI*: 10.1109/TGRS.2016.2565471*, 2016.

PII Matti Eskelinen. Software Framework for Hyperspectral Data Exploration and Processing in MATLAB. *ISPRS SPEC3D 2017 : Frontiers in Spectral imaging and 3D Technologies for Geospatial Solutions (pp. 47-50). International archives of the photogrammetry, remote sensing and spatial information sciences, Volume XLII-3/W3. International Society for Photogrammetry and Remote Sensing.* DOI*: 10.5194/isprs-archives-XLII-3-W3-47-2017*, 2017.

PIII Leevi Annala, Matti Eskelinen, Jyri Hämäläinen, Aamos Riihinen, Ilkka Pölönen. Practical Approach for Hyperspectral Image Processing in Python. *ISPRS TC III Mid-term Symposium "Developments, Technologies and Applications in Remote Sensing" (pp. 45-52). International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-3. International Society for Photogrammetry and Remote Sensing.* DOI*: 10.5194/isprs-archives-XLII-3-45-2018*, 2018.

PIV Roberts Trops, Anna-Maria Hakola, Severi Jääskeläinen, Antti Näsilä, Leevi Annala, Matti Eskelinen, Heikki Saari, Ilkka Pölönen, Anna Rissanen. Miniature MOEMS hyperspectral imager with versatile analysis tools. *Proceedings of SPIE Volume 10931 : MOEMS and Miniaturized Systems XVIII; 109310W (pp. 109310W). SPIE conference proceedings, 10931. SPIE, The International Society for Optical Engineering.* DOI*: 10.1117/12.2506366.*, 2019.

PV Matti Eskelinen, Jyri Hämäläinen. Dangers of Demosaicing : Confusion From Correlation. *WHISPERS 2018 : 9th Workshop on Hyperspectral Image and Signal Processing, Evolution in Remote Sensing. IEEE.* DOI*:10.1109/WHISPERS.2018.8747204*, 2019.

In article **PI** the author carried out the moisture estimation from the spectra, visualised the results and wrote the respective sections of the original manuscript.

The author conceptualized and wrote the software presented in article **PII**, created the presented examples and wrote the original manuscript.

For article **PIII**, the author collaborated with Leevi Annala and Jyri Hämäläinen in the investigation of existing software tools and conceptualisation of the presented methods. The author also wrote the original algorithm for the histogram visualisation and gave comments on the original draft of the manuscript.

The author developed the `fpipy` software library used in articles **PIV** and **PV** in collaboration with Jyri Hämäläinen. The author also supervised and contributed to the development of the `pyspindl` library developed by Aamos Riihinen and used in articles **PIII** and **PIV**, the software libraries `spectracular` and `camazing` developed by Severi Jääskeläinen and used in article **PIV**, and contributed to the CubeView software developed by Anna-Maria Hakola and used in article **PIV**.

The research question and methodology of article **PV** was posed by the author. The software and manuscript were written in collaboration with Jyri Hämäläinen, who carried out the main investigation.

1 INTRODUCTION

1.1 Background

The light that reaches our eyes and presents us with a view of the world around us is the result of countless interactions between the matter and photons, the particles that transmit the electromagnetic force. Photons are mainly created by the movement of electric charge — electrons or protons — in various ways, whether in the form of current flowing through a conductor or an electron dropping from an energy level to another as it zips around the nucleus of an atom, each event resulting in a photon of a wavelength corresponding to the lost energy. The multitudes of ways photons can be born give rise to the full electromagnetic spectrum, containing everything from gamma rays and radio waves to the photons of light that we detect as colours in the world around us, as well as the ultraviolet and infrared we do not. Ordered by wavelength, the range of visible light extends approximately from 400 to 700 nanometers as depicted in figure 1.

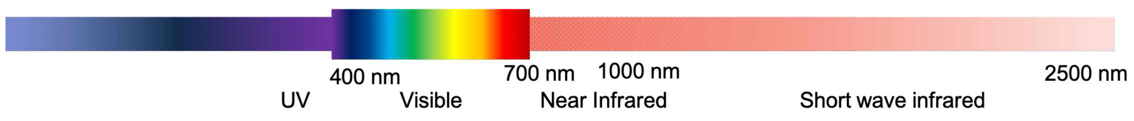


FIGURE 1 The wavelength range of light. Image from Ilkka Pölönen.

As photons interact with matter, they are absorbed, transmitted and reflected in various ways depending on the substances and wavelengths of the photons involved. This interaction is what results in us perceiving different materials as having different colours, opacities and reflectivities, as our eyes and brains interpret the distribution of photons received from different surfaces. However, our eyes not only discern just a fraction of the electromagnetic spectrum, they reduce the billions of different photons to mixes of just three colours, losing much of the information carried by the photons of the processes that created them. The same is true of most cameras, as they are built to imitate the human vision.

Spectral imaging is the combination of spectroscopy and imaging, the first

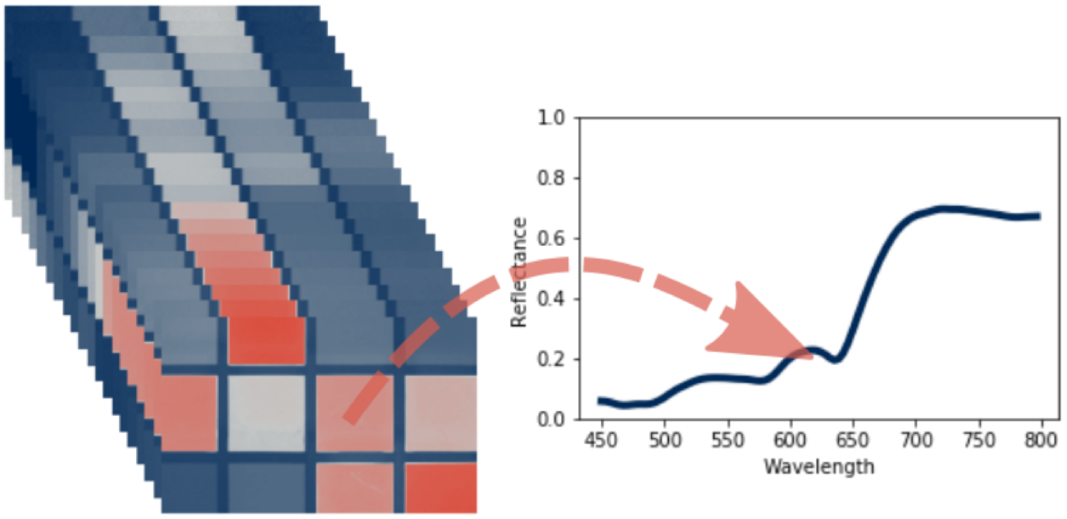


FIGURE 2 Left: Stack of images. Right: A wavelength spectrum. Image from Ilkka Pölönen.

being the study of information in the wavelength (or energy) domain and the latter in the spatial domain. By finding ways to control the passage of light through the camera based on its wavelength, the full spectrum and the information it contains can be recorded and analysed. A way to enact such separation of white light into it's component wavelengths by using a prism was first discovered by Newton and published in his treatise ”Opticks” in 1704. This finding eventually lead to the understanding of the properties of light that we have today, and since then many more methods have been devised to better study light. The methods of capturing images for later use have also evolved from exposing expensive and single-use silver-halide films that need to be analysed by hand to using photoelectric materials to record electronic signals that can be directly passed to a computer for automated processing, leading to the development of miniaturized cameras that enable high-resolution photography with the press of a button. (Nakamura, 2006)

Using three filters with different spectral responses to capture a colour image is the most rudimentary form of spectral imaging. Increasing the number of filters and narrowing their spectral responses leads to multispectral imaging, with the filters usually precisely tuned to give the most information about the phenomena under study. The most flexible imaging devices to date are hyperspectral cameras (or imaging spectrometers), which can capture and separate light into tens or hundreds of bands, forming a nearly continuous spectrum at each pixel. During the past half-century, multiple types of spectral imagers have been developed using different techniques. These can be roughly classed to spatially scanning, spectrally scanning and snapshot cameras.

So-called pushbroom and whiskbroom imagers imitate Newton's methods of observing the spectrum by using a slit to select a narrow beam of light that is passed to a prism or grating, which then disperses the light to different parts of the focal plane of the sensor based on its wavelength. The image is formed by scanning the aperture of the imager across the imaged subject, either by moving the imager or the subject. These types of imagers have been used for remote sensing studies of the Earth since 1980's using airplane-mounted sensors such as Jet Propulsion Laboratory's AVIRIS (Vane et al., 1993) or recently in satellites, for example the MSI on the European Space Agency's Sentinel-2 mission (Drusch et al., 2012).

Exchangable and tunable filters allow the capture of all the spatial information at a single wavelength at a time, but need to capture each wavelength at a separate point in time. While electronically tunable filters have allowed much faster imaging then rotating filter wheels, existing designs have still been relatively slow and low resolution (Gat, 2000). Snapshot cameras that can capture both the spatial and spectral information in a single exposure have also been developed using clever optical and computational methods, but they likewise are generally very costly to manufacture or have low resolution (Gao and Wang, 2016).

A recently developed method for producing a more versatile tunable filter is based on the use of electrically tunable Fabry–Perot interferometer (FPI). Imagers based on this technology have been built by VTT Technical Research Centre of Finland, first with piezo-actuated and later using MEMS FPI devices. The combination of these filters with colour filter array sensors allows the capture of multiple narrow bands with a single exposure, allowing for faster spectral scanning than existing tunable filter designs (Saari, Aallos, et al., 2009; Saari, Pölönen, et al., 2013). The degree of miniaturization and ability to use commoditised imaging components has made the technology an attractive and affordable choice for some novel applications such as nanosatellites (Mannila et al., 2013).

1.2 Research environment

The author's research has been carried out as part of the activities of the spectral imaging laboratory of the Faculty of Information technology, headed by Dr. Ilkka Pölönen. These activities have included application-focused work using data from Fabry–Perot interferometer colour cameras, either from collaborators outside the university or gathered using the VTT-provided spectral imagers housed at the laboratory. The application studies (especially the less successful ones) have then prompted investigations into the computational characteristics of the imagers, leading to the questions and results presented in this thesis.

The main hardware the author has been working with are prototype FPI colour cameras manufactured by VTT, one using a piezo-actuated FPI device (figure 3, left) with a PointGrey (now FLIR) machine vision camera and another with a MEMS FPI device (figure 3, right) combined with a Basler machine vision camera. The latter is shown assembled with the camera and optics on the left in figure 7 in section 2.4. Both FPI devices are controlled through a serial interface by passing in a voltage value to set the FPI air gap. The machine vision cameras are both GenICam compatible with RGB Bayer filters, though with different patterns.

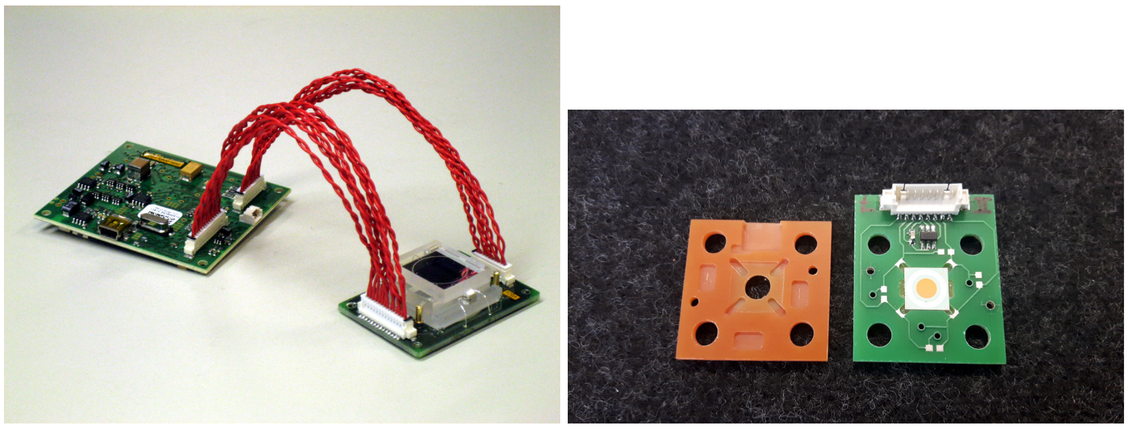


FIGURE 3 Fabry-Perot interferometer devices. Left: Piezo-actuated FPI device with controller. Right: MEMS FPI device. Images from Heikki Saari, VTT.

Originally the first prototype camera was only usable through the VTT-provided Labview interface and additional post processing and analysis were performed using Matlab. As research progressed, the deficiencies and closed nature of both platforms were felt to be an obstacle both for the efficiency and accessibility of research. This motivated a move to develop a new set of tools using the Python programming language, as it was both open and freely available, and had gathered a good ecosystem of machine learning libraries which we wished to be able to utilise. The migration to Python has also allowed for more structured software development than either Matlab or Labview, which is evident in the libraries presented in this work.

1.3 Main research questions

This thesis and the included articles answer the following questions that have arisen during the author working with hyperspectral cameras using the Fabry–Perot interferometer as their wavelength discriminating element:

1. What data is required to compute spectroscopic radiances from a Fabry–Perot imaging system with a colour camera back-end?
2. Can the process be implemented in a general purpose computing environment?
3. Is data from such imagers usable for hyperspectral analysis?

1.4 Outline of the thesis

Chapter 1 provides a short introduction to the history of hyperspectral imaging and introduces the thesis.

Chapter 2 of the thesis provides a short overview of the physics and technology of colour cameras and that of Fabry–Perot interferometers when used for wavelength filtering. This is provided to establish conventions and serve as reference material for the later parts of the thesis.

Chapter 3 of this thesis presents a short overview of the results of the included articles and the software libraries published during the research.

Chapter 4 extends the included articles with a detailed description of the steps needed for the acquisition and post-processing of hyperspectral data using FPI colour camera imagers and documents the algorithms and solutions to computational problems that are implemented in the `fpipy` library.

Finally, chapter 5 presents conclusions drawn from the research and discusses the future of hyperspectral imaging research using FPI imagers.

2 THEORY AND BACKGROUND

This chapter will guide the reader through the theory of radiometry, digital imaging and Fabry–Perot interferometers to highlight the computationally relevant characteristics of the imager components and the data that can be gathered using them.

2.1 Radiometry

Radiometry is the study of radiation, and as such covers far too wide a breadth of human knowledge to be condensed in a section of this thesis. Defining the quantities that are being measured by the imaging system is however useful, and for that purpose the relevant definitions of spectral radiance and reflectance are presented here following those of Nicodemus et al., 1977 and Manolakis, 2016. For a comprehensive view of the definitions of the geometry of reflectance measurements, one should consult the review of confusions about reflectance given by Schaepman-Strub et al., 2006.

Wavelength in the context of radiometry refers to the wavelength of the photons or light in question and is denoted by the Greek letter λ. In the context of hyperspectral imaging, these usually fall into the visible and near-infrared (VNIR) or short-wave infrared range (SWIR) of the spectrum and explicit values are usually reported in units of nanometres.

The fundamental quantity of study in radiometry is the radiance, denoted by L and defined as the radiant flux (power) per unit projected area per unit solid angle. Given a radiant flux $\Phi(x, y, \theta, \phi)$ in the direction of (θ, ϕ) on the sphere on or from the point (x, y) on the surface, the pointwise radiance can be expressed as a derivative of the flux as

$$L(x, y, \theta, \phi) = \frac{\mathrm{d}^2\Phi(x, y, \theta, \phi)}{\mathrm{d}A_{\text{proj}}\,\mathrm{d}\omega}, \tag{1}$$

with

$$\mathrm{d}A_{\text{proj}} = \mathrm{d}A\cos(\theta) \quad \text{and} \tag{2}$$

$$\mathrm{d}\omega = \sin\theta \mathrm{d}\theta \mathrm{d}\phi \tag{3}$$

the differential projected area in the direction of the flux and the differential solid angle, respectively.

The definition of spectral radiance follows by substituting the flux in definition 1 with the spectral flux (or power per wavelength) $\frac{\mathrm{d}\Phi}{\mathrm{d}\lambda}$, giving

$$L_\lambda(x,y,\theta,\phi,\lambda) = \frac{\mathrm{d}^3\Phi(x,y,\theta,\phi,\lambda)}{\mathrm{d}A_{\text{proj}}\,\mathrm{d}\omega\,\mathrm{d}\lambda} = \frac{\mathrm{d}L(x,y,\theta,\phi,\lambda)}{\mathrm{d}\lambda}. \tag{4}$$

The relevant quantity for determining surface properties is the bidirectional reflectance factor R, which is a dimensionless quantity defined as a ratio of the reflected radiant flux Φ_r from the surface area $\mathrm{d}A$, and the reflected flux Φ_r^{id} of an ideal (non-absorptive and -transmissive) diffuse reflector of the same area. Denoting the incoming flux direction with (θ_i, ϕ_i) and the reflected as (θ_r, ϕ_r) gives the definition as

$$R(x,y,\theta_i,\phi_i;\theta_r,\phi_r) = \frac{\mathrm{d}\Phi_r(x,y,\theta_i,\phi_i;\theta_r,\phi_r)}{\mathrm{d}\Phi_r^{id}(x,y,\theta_i,\phi_i)} = \frac{\mathrm{d}L_r(x,y,\theta_i,\phi_i;\theta_r,\phi_r)}{\mathrm{d}L_r^{id}(x,y,\theta_i,\phi_i)}. \tag{5}$$

The corresponding spectral quantity can be derived in the same fashion as for the radiance.

The reflected flux from the ideal reflector is only dependent on the direction of the incoming light, but a non-diffuse surface under study can have the reflected flux vary with respect to the viewing direction. Reflectance factors can thus reach values greater than one for some angles and surfaces. This fact is a cause of confusion as the measured reflectance factors are often conflated with the reflectance ρ, which is defined as a ratio of the reflected and incident fluxes and as such can never exceed one for a non-emitting surface.

Since they are defined using infinitesimal areas and angles, real measurements can never determine these quantities directly (Nicodemus et al., 1977). Instead, integral quantities are measured and the pointwise quantities are estimated based on knowledge of the measuring geometry and other factors. However, this difference is not reflected in the use of language and is commonly overlooked in the literature.

2.2 Digital image sensors

A digital image sensor is composed of an array of photosensitive elements that are exposed to light and convert the photons into charge. These are usually pinned photodiodes (Fossum and Hondongwa, 2014), each of which is referred to as a pixel. The charges are converted to voltage, which then passes to an analog-to-digital converter (ADC) which converts it to numbers, referred to as Digital

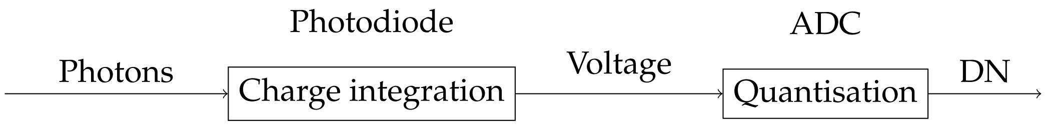


FIGURE 4 Diagram of the operation of a digital imaging sensor.

Numbers (DN) or Analog-to-Digital Units (ADU). Figure 4 shows a diagram of the operations.

The quantum efficiency of a sensor element is the ratio of the number of electrons produced and the number of photons impacting the element. It is generally a function of wavelength due to physical properties of the detector material. When considering colour sensors with a per-pixel varying coating in this thesis, the quantum efficiency is also considered to include the transmission of the coating to follow (Saari, Pölönen, et al., 2013). Current sensors are generally based on either CCD (Charge Coupled Device) or CMOS (Complementary Metal-Oxide-Semiconductor) technology with assisting circuitry, which differ in the way in which they are manufactured and the way charges are collected and eventually converted to numbers. The type of the sensor does not influence the computational features of the data in the scope discussed in this thesis, but interested readers may refer to (Nakamura, 2006) for a comprehensive overview of differences between the sensor types and other factors related to digital imaging. Still, There are computationally relevant values that do need to be known in order to compute radiometrically correct data.

In cameras with mechanical shutters, exposure time refers to the length of time the film or sensor is exposed to incoming light by physically obstructing the optical path with the shutter for a set period. In digital sensors, the charges that accumulate in the photosensitive elements during the exposure are always collected after a set period of time after which the elements are reset to a chargeless state. This method of exposure is referred to as an electronic shutter, and can be used either by itself as the sole method of control the amount of charges measured, or in tandem with a mechanical shutter to allow separate control over the light exposure and electronic collection.

The gain of a sensor is a value that relates the number of photons that have hit the photodiode to the generated DN value output of the analog-to-digital converter during quantization of the signal. In order to generate a given value, the number of electrons collected in the photodiode must first be converted to a voltage between a minimum and a maximum voltage acceptable by the ADC. The ADC then maps this range of voltages to a set of discrete numbers of a given bit depth B, resulting in a DN value between 0 and $2^B - 1$. Depending on the camera, the gain setting controls either the voltage generated by a given number electrons or the output of the ADC, respectively referred to as analog or digital gain. If the camera gain is user-adjustable, the relation between the adjustable gain setting and the output values should be determined.

In addition to the signal generated by the sensor for a given number of photons, the electronics introduce noise to any given measurement. The noise is generally composed of so-called hot and cold pixels caused by defects in the sensor, read-out noise caused by electronic interference during collection of the charges, and temperature and exposure-time dependent noise. In laboratory settings, a common way to correct for this noise is to take a number of frames with the aperture of the imager covered with an opaque material (often just a lens cap) and compute a pixelwise median or mean of the signal over these exposures, which is then subtracted from the measured signal. This is referred to as dark subtraction or dark correction, though the latter term also more generally applies to methods for estimating and correcting for noise.

2.3 Colour filter arrays

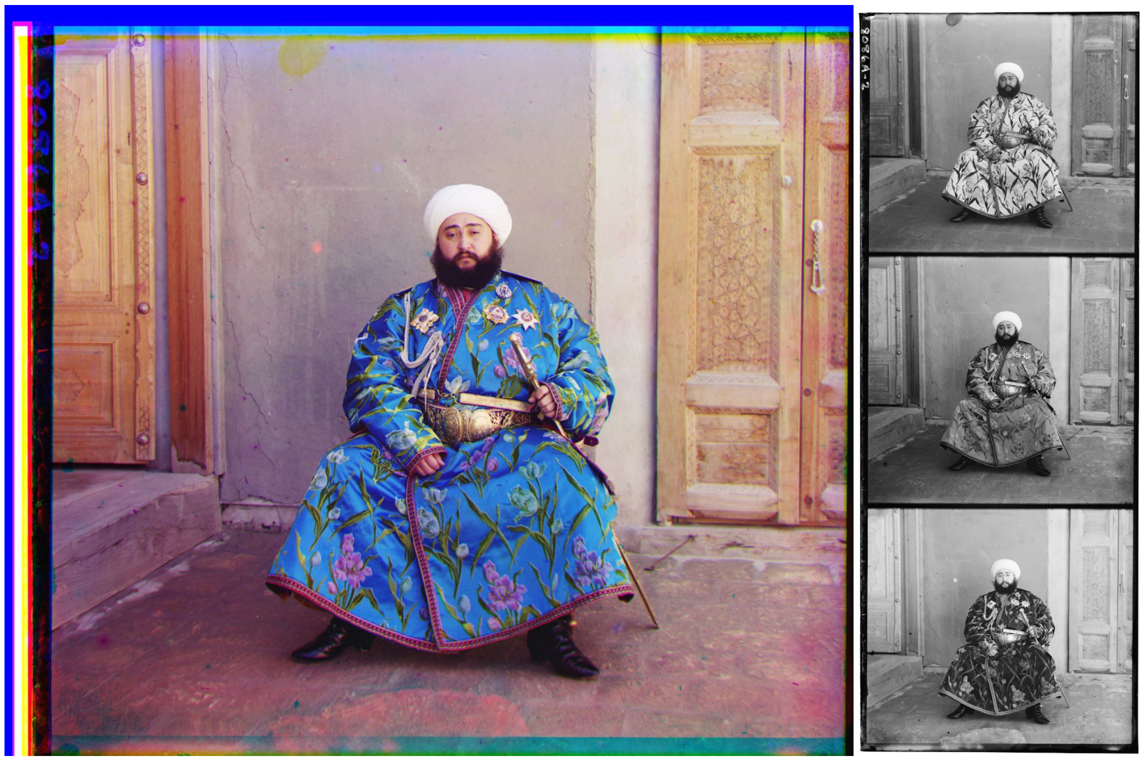


FIGURE 5 A composite colour image taken by Sergei Mikhailovich Prokudin-Gorskii in 1911 composed of three separate exposures (shown on the right). Image from the digital collection of the Library of Congress.

A monochromatic sensor does not allow for differentiation of photons of different wavelengths, only counting the total number of photons that cause electrons to be accumulated in the detector material. Adding a coloured filter in front of a photodiode allows that pixel to only detect the wavelengths passing through the filter. Early color cameras functioned in this exact manner (see figure 5): By taking images with three different filters, one could use the three monochrome images with suitable pigments to reconstruct a color image. However, changing the filter takes time, during which the subject needs to stay still or else the three images will not match up.

One solution modern cameras use to solve this is the use of a Bayer filter, a technology patented by Bayer in 1976. Instead of placing a separate filter over the monochromatic sensor, different filters are placed over the pixels of the sensor in a set pattern. Patterns composed of 2×2 pixels of red, green and blue pixels are referred to as Bayer patterns. The full tiling of the sensor is called the Bayer filter or Bayer filter array.

Examples of commonly used patterns are presented in figure 6. As is evident from the patterns, the green component is usually sampled with twice the frequency of the red and blue components. This stems from the colour green being most influential in determining luminance in human vision. As the aim of most colour camera manufacturers is to capture colours as they are captured by the human eye, the necessity of accurate capture of the green component is magnified.

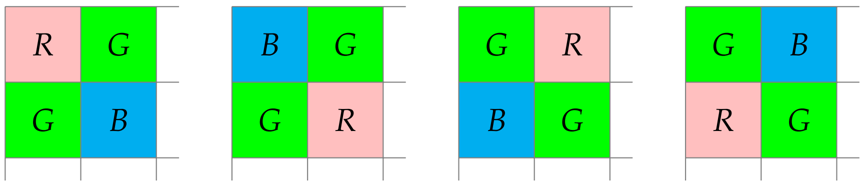


FIGURE 6 Bayer filter patterns. The patterns are identified using strings denoting the colors of the 2×2 pattern as read from left to right and top to bottom. The 2×2 pattern is repeated across the sensor. From the left: RGGB, BGGR, GRBG and GBRG patterns.

Patterns with more complicated arrangements and other colours of pixels also exist. There have also been array types developed specifically for spectral imaging using more pixel types to achieve better spectral discrimination, such as presented by Lapray et al., 2014.

In order to reconstruct an RGB image with all three values in each pixel, interpolation is used to estimate the missing colours in the pixels in which they are not measured. Arguably the simplest way to interpolate in a 2D plane is to compute the average of the neighbouring pixels of a given colour at each missing point. This is referred to as bilinear interpolation, as it is indeed a linear approximation of the missing values in two dimensions. The linear nature of the interpolation is useful, as it turns out that it can be computed as a discrete convolution of the separated colour planes R', G' and B with zeros in place of each missing value as

$$\begin{aligned} R &= K_{\mathrm{RB}} * R' \\ G &= K_{\mathrm{G}} * G' \\ B &= K_{\mathrm{RB}} * B' \end{aligned}$$

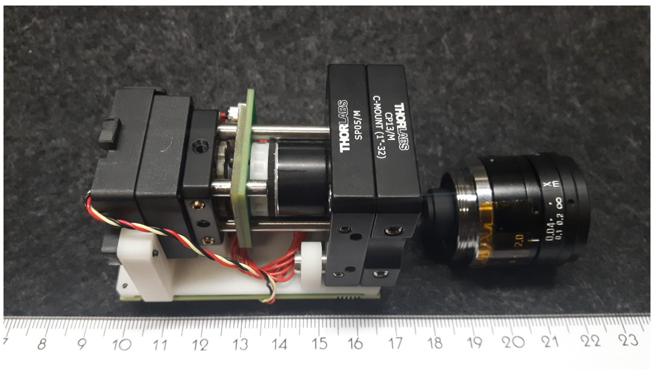
with the kernels

$$K_{\mathrm{G}} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{bmatrix} /4 \quad \text{and} \quad K_{\mathrm{RB}} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} /4 \tag{6}$$

that take into account the different rate of sampling between the green and red or blue components.

2.4 The Fabry–Perot interferometer

A Fabry–Perot interferometer (or etalon) consists of two parallel half-mirrors spaced close together. A beam of light entering the etalon interferes with itself as it reflects off the mirrors, causing integer multiples of certain wavelengths to be amplified and transmitted through the mirrors. This allows the use of the interferometer as a highly sensitive instrument for measuring thicknesses by observing the interference fringes or examining fine spectral features, as originally proposed by (Perot and Fabry, 1899), or conversely as a narrow-band wavelength filter by fine control of the mirror distances given some supporting optics (as utilized by Saari, Pölönen, et al., 2013).



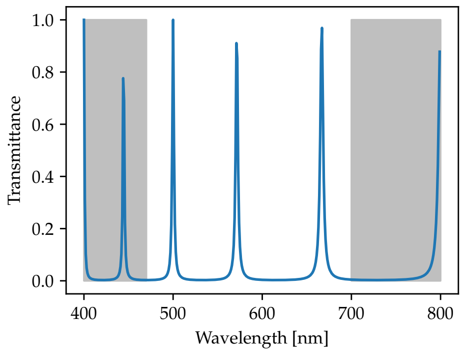


FIGURE 7 Left: An assembled Fabry–Perot interferometer colour camera. The green circuit board is a MEMS FPI device, with the camera to the left and lenses and filters to the right. Image by Ingmar Stuns (VTT). Right: Simulated FPI transmittance between 400 and 800 nm for an air gap of 2 µm and mirror reflectivity of 0.9. The grey area shows wavelengths that must be blocked by low- and highpass filters to prevent more than 3 peaks reaching the sensor at once.

The transmittance of the Fabry–Perot etalon as a function of wavelength λ and a mirror gap length d is approximately

$$T_{\mathrm{FPI}}(\lambda, d) = \frac{T_{\mathrm{m}}(\lambda)^2}{1 + R_{\mathrm{m}}(\lambda)^2 - 2R_{\mathrm{m}}(\lambda)\cos\left(\frac{4\pi d\cos(\theta)}{\lambda}\right)}, \tag{7}$$

with T_{m} and R_{m} the transmittance and reflectance of the mirrors and θ the angle of incidence of the incoming light with respect to the optical axis of the mirrors. Figure 7 shows a simulated transmittance of the FPI in the visible range for a single air gap.

By introducing a low- and high-pass filters into the optical path (see figure 7) the set of transmittance peaks of the FPI through which the light passes on to the sensor can be limited to three or less for a range of mirror distances. This allows the modeling of the pixel responses for each gap d and interference order n of the FPI as

$$S_{\mathrm{R}}(d,n) = \int_{\lambda_{\min}(d,n)}^{\lambda_{\max}(d)} \eta_{\mathrm{R}}(\lambda) T_{\mathrm{FPI}}(\lambda,d) T_{\mathrm{sys}}(\lambda) \mathrm{d}\lambda \tag{8}$$

$$S_{\mathrm{G}}(d,n) = \int_{\lambda_{\min}(d,n)}^{\lambda_{\max}(d)} \eta_{\mathrm{G}}(\lambda) T_{\mathrm{FPI}}(\lambda,d) T_{\mathrm{sys}}(\lambda) \mathrm{d}\lambda \tag{9}$$

$$S_{\mathrm{B}}(d,n) = \int_{\lambda_{\min}(d,n)}^{\lambda_{\max}(d)} \eta_{\mathrm{B}}(\lambda) T_{\mathrm{FPI}}(\lambda,d) T_{\mathrm{sys}}(\lambda) \mathrm{d}\lambda \tag{10}$$

respectively for the R, G and B pixels, with η the quantum efficiency of the respective pixel and T_{sys} the transmittance of the rest of the optical system without the FPI. The value $\lambda_{\min}(d,n)$ is the midpoint between the transmittance orders n and $n-1$, and conversely $\lambda_{\max}(d,n)$ is the midpoint between the orders n and $n+1$. The center wavelengths of each order can be approximated as

$$\lambda_n = \frac{2d}{n}, \tag{11}$$

allowing the midpoints to be computed from the approximation. (Saari, Aallos, et al., 2009; Saari, Pölönen, et al., 2013)

For a fixed FPI air gap d, the combined RGB signals can be modeled using the signal responses as a linear combination of three consecutive transmission peak signals $S_n = S(d,n)$ as

$$\begin{bmatrix} S_{\mathrm{R}} \\ S_{\mathrm{G}} \\ S_{\mathrm{B}} \end{bmatrix} = \begin{bmatrix} S_{Rn+2} & S_{Rn+1} & S_{Rn} \\ S_{Gn+2} & S_{Gn+1} & S_{Gn} \\ S_{Bn+2} & S_{Bn+1} & S_{Bn} \end{bmatrix} \begin{bmatrix} S_{n+2} \\ S_{n+1} \\ S_n \end{bmatrix}. \tag{12}$$

Formally, the unknown radiances can be solved from this equation by multiplying both sides of the equation by the inverse of the sensor response matrix. Using the calibration procedure outlined by Saari, Aallos, et al., 2009, the coefficients S_{Xn}^{-1} of the inverse transform[1] can be empirically determined. These allow the recovery of the unknown radiances given RGB measurements R, G and B as

$$S(n) = R \cdot S_{\mathrm{R}}^{-1}(n) + G \cdot S_{\mathrm{G}}^{-1}(n) + B \cdot S_{\mathrm{B}}^{-1}(n) \tag{13}$$

As the calibration coefficients S_{Xn}^{-1} depend on the signal output of the imager during the calibration procedure, their scale is set by the imaging parameters used during the calibration, if not normalized afterwards. These include gain, exposure time and binning and any other modifiable parameters which can affect the overall signal level output of the imager for a given radiance. As such, the recovered signal cannot be considered to be radiometrically correct radiance without explicit corrections if the parameters used during imaging do not correspond with the calibration. Given linearity of the sensor response (which should preferably be

[1] S_{Xn}^{-1} denoting the element of the inverse matrix in the same position as S_{Xn} in eq. 12

calibrated rather than assumed), the uncorrected values should still be usable for computing reflectances or transmittances as direct ratios if the parameters between the subject and reference images are the same.

3 OUTLINE OF RESULTS

This chapter presents an overview of the main results of the research in the included papers and a description of the main software libraries that have been developed by the author during the thesis project.

3.1 PI: Remote Sensing of 3-D Geometry and Surface Moisture of a Peat Production Area Using Hyperspectral Frame Cameras in Visible to Short-Wave Infrared Spectral Ranges Onboard a Small Unmanned Airborne Vehicle (UAV)

This article evaluated the use of an FPI colour sensor imager in the estimation of peat moisture over large areas using a drone camera platform. For this article the author performed the moisture analysis starting from the measured spectral reflectances and ground truth data on the moisture. The analysis included bandwise correlation analysis, support vector machine (SVM) training and evaluation and visualization of the results, which were carried out using Matlab. The FPI data showed good correlation with the moisture content using the expected wavelengths. The SVM analysis was able to predict peat moisture on average to within 10 percent of the actual value, demonstrating the feasibility of using a drone-mounted imager to provide usable data for machine learning applications. Experience from the study provided the motivation for the exploration and development of the software tools presented in later papers, as considerable programming effort was needed to carry out and visualise results of the analysis using the available tools.

3.2 PII: Software Framework for Hyperspectral Data Exploration and Processing in MATLAB

This article introduces the `hsicube` library developed by the author for the exploratory analysis of hyperspectral images using a high-level language. The presented software tools are a direct consequence of the frustration experienced with the existing tools, aimed at reducing the mental effort needed in the development of hyperspectral analysis algorithms by automating common operations and providing abstractions for chaining workflow tasks. This is demonstrated in the paper by comparison of the code needed to perform these tasks using existing Matlab methods and the new object-oriented methods.

3.3 PIII: Practical Approach for Hyperspectral Image Processing in Python

Article **PIII** expands upon **PII** by exploring ways to achieve the same analysis workflow using existing software tools in a general purpose, high-level programming language. The presented tooling was evaluated by the authors during a lab-wide push to switch programming environments and found to be a good basis for further development, considering availability of existing tools and expected development effort to convert workflows. The presented exploration workflows and the implementation of the spectral index library `pyspindl` demonstrated that workflows such as the one in **PI** could be achieved with less effort than with existing tools.

3.4 PIV: Miniature MOEMS hyperspectral imager with versatile analysis tools

This article presents miniaturized spectral imaging hardware and a software framework for acquiring and analyzing hyperspectral imagery from FPI imagers using a general-purpose programming language. The software framework is built around the data structures and tools presented in article **PIII**, expanded with tools for hardware control of cameras and FPI devices respectively with the `camazing` and `spectracular` libraries, and the user facing CubeView software. The main contribution of the author is the `fpipy` library, which is used by the FPI control software to determine the calibrated FPI gap lengths for a given image, and the CubeView software for performing the interpolation and computing radiances and reflectances from the raw data. The combined software stack demonstrated the ability for the framework to capture, process and analyze hyperspectral images from a novel sensor that is also easily extendible with new algorithms and allows interoperation with existing machine learning tools.

3.5 PV: Dangers of Demosaicing : Confusion From Correlation

In this article we demonstrated a potential source of bias in the interpolation methods used to process raw data from colour cameras when used together with FPI filters. For this article the author collaborated with Jyri Hämäläinen to build a simulation framework for assessing the result quality of different interpolation methods as used in the `fpipy` library. The findings indicated that demosaicing methods intended for use with normal colour cameras can introduce wavelength-dependent errors when used with FPI hardware. This study motivated the development of the algorithm presented in section 4.3 as part of the workflow, and demonstrates concerns that need to be taken into account when using existing image processing algorithms for hyperspectral images.

3.6 `hsicube` Matlab package

The `hsicube` Matlab package presented in **PII** is a collection of tools for hyperspectral analysis. The central piece of the package is the Cube data type which allows object-oriented handling of hyperspectral data including wavelength information and other metadata that without it would have to managed by juggling multiple arrays and indices manually. The Cube class also includes algorithms for content-aware visualizations and thresholding. The package is available under the MIT license at github.com/silmae/hsicube.

3.7 `fpipy` Python library

The `fpipy` library contains Python implementations of algorithms that are needed to process a dataset of raw camera frames and metadata into spectral radiances. It utilises a number of existing open source libraries for most of the heavy lifting, and combines them with implementations of the hyperspectral processing detailed in sections 4.2, 4.3 and 4.4. It also includes programmatically available conventions for the spectral data structures, which are used to provide consistent and testable data handling from image acquisition back-ends to user-facing applications. The software is available under the MIT license at github.com/silmae/fpipy.

4 HYPERSPECTRAL IMAGING USING A FABRY–PEROT INTERFEROMETER AND COLOUR FILTERS

This chapter provides an overview of the computational and workflow steps needed for efficient acquisition and post-processing of hyperspectral data using an imager composed of a Fabry–Perot interferometer and a colour filter array sensor.

4.1 Image acquisition

In order to acquire hyperspectral data using the VTT FPI colour camera imagers three software components were developed during the research by the author and coauthors of article **PIV**:

1. Programmable camera interface (`camazing`)
2. FPI controller interface (`spectracular`)
3. Post-processing software for computing radiances (`fpipy`)

The manufacturer calibration of the FPI device determines the set of gap lengths that need to be scanned in order to capture a given hyperspectral data cube. During imaging, the FPI controller library sets the FPI to a given air gap and passes control to the camera interface, which then returns the image and other gathered data to the controlling software. The FPI is then set to the next position for the capture of the next frame, and the process is repeated until all selected FPI settings have been imaged.

The image needs to be accompanied by a set of metadata in order for the post-processing software to be able to reconstruct the spectral radiances. Table 1 lists all the necessary data with typical types and dimensions that need to be passed to the post-processing software in order to produce a single frame of radiance information with all the data listed in table 2.

The software components utilize the `xarray` Python library (Hoyer and Hamman, 2017) to collect the data into a dataset that can be passed around in memory or serialised to disk while containing all the necessary data in a single variable or file. Of the necessary data, the inversion coefficients used for radiance calculation must be first determined for the device using the calibration procedure described in section 2.4. If the device allows different passband filters to be used to select the range of imaged wavelengths, care must be taken to use the calibration information corresponding to the filter in use.

TABLE 1 Data for computing spectral radiance given a single air gap with expected types and dimensions for numerical arrays. N_{peaks} is the number of FPI orders between the low- and high-pass filters, and N_{colours} the number of colours present on the Bayer filter matrix. N_{x} and N_{y} are the number of pixels in the given dimension of the sensor.

Data field	Type	Dimensions
N_{peaks}	int	1
N_{colours}	int	1
Inversion coefficients (S_{Xn}^{-1})	float	$N_{\text{peaks}} \times N_{\text{colours}}$
Peak wavelengths	float	N_{peaks}
Peak widths	float	N_{peaks}
Bayer pattern	string	-
Pixel format	string	-
Exposure time	float	1
Gain value	float	1
Raw image data	unsigned int	$N_{\text{x}} \times N_{\text{y}}$

TABLE 2 Radiance data computed from a single frame with a given FPI air gap. The dimensions N_{peaks}, N_{x} and N_{y} correspond to those in table 1.

Data field	Type	Dimensions
Peak wavelengths	float	N_{peaks}
Peak widths	float	N_{peaks}
Radiance bands	float	$N_{\text{peaks}} \times N_{\text{x}} \times N_{\text{y}}$

When using a GenICam (EMVA, 2019) compatible camera as the imaging backend, the Bayer pattern, pixel format, gain and exposure time can be queried using the GenICam API. The gain value and exposure time should in general be stored for each frame, even if they are constant during the cube capture. This allows data dimensions to be consistent and simplifies processing of the data, as no ragged arrays are needed. Similarly, if the number of peaks is not constant for all the settings in the set of air gaps, the inversion coefficients, peak wavelengths and peak widths should be stored in arrays with size $\max_d N_{\text{peaks}}$ instead of the settings-specific number of peaks to enable the complete set to be stored as a non-ragged array.

If the camera interface is set to decode the camera native pixel format to a specified range instead of storing the DN values directly to a possible larger container format, the pixel format information should be retained so the actual range and packing of the data can be determined. Alternatively, the information can be stored explicitly in some format, for instance using the NetCDF attribute conventions to indicate data value scaling.

In order to compute reflectance or transmittance factors, reference images of the reflectance standard or illumination should be taken using the same FPI settings. If only the rational quantities are needed for the analysis, gain and exposure information is not strictly necessary for their measurement, as they cancel out when computing the ratios (assuming that the reference and subject frames use the same gain and exposure).

4.2 Dark correction

As detailed in section 2.2, the output of the digital sensor may be nonzero even when no light from the aperture is allowed to hit it due to thermal and electronic noise. Without correction this noise will introduce bias to any computed quantities, especially in the case of FPI colour camera imaging since the computed radiances are highly sensitive to the ratios between the RGB component measurements. Given a frame I from the imager and an estimate D of the dark current (including all additive noise components), the true signal is in principle computed as $I - D$. This simple subtraction can however introduce computational complications, if the estimate violates either of the following assumptions.

1. $D \leq I$,
2. D is of the same numeric type as I.

Averaging dark frames to produce an estimate of the dark current can easily produce estimates which violate both assumptions depending on the implementation of the mean or median used. Due to the inherent randomness of the dark current, it is likely that dark pixels in the actual illuminated frames will contain values lower than the estimate, and taking either the mean or the median of multiple values can cause the estimate to be represented as either floating point or larger integer types than the original frames. Breaking the first assumption forces the choice of either clamping the result values to nonnegative ones or allowing the result to contain negative values.

The first option is sensible given the nature of imaging, as negative values are clearly unphysical. It must then however be taken into account when building a model of the imager performance, since clamping complicates the statistical description of noise actually subtracted from the data. Clamping is currently implemented in `fpipy` by multiplying each value with a boolean mask with each

```
def subtract_clamp(x, y):
    return (x > y) * (x - y)
```

LISTING 4.1 Zero-clamping dark subtraction implementation using `numpy`. Original implementation courtesy of Sampsa Kiiskinen.

using `numpy` (listing 4.1). This prevents integer rollover while not forcing a cast of the data to a larger signed integer or floating-point type, which would be undesirable. However, upcasting of the results still occurs if the either parameter is of a larger type, which ensures no precision is lost due to forced downcasting.

The other option of allowing negative values would also force casting the result to either a larger signed integer type or floating point with the associated memory cost. This also causes problems for post processing steps if they assume nonnegativity of the data, which for the author resulted in many interesting hours of debugging before the implementation of clamping. Given the possible problems that integer rollover or negative values can cause, it is surprising that the issues are not mentioned in descriptions of dark correction methods in either textbooks (Manolakis, 2016, Section 1.4.1) or articles on radiometric corrections (Minařík, Langhammer, and Hanuš, 2019, Section 2.3.2).

4.3 Demosaicing

Demosaicing (i.e. interpolation) of the Bayer mosaic in some form is necessary for the recovery of the radiance signal from imagers using colour sensors with tunable filters. In contrast to monochromatic sensors for which bandwise reflectances or transmittances can be computed directly by division without converting the DN values to radiances, mixed-peak radiances must first be separated. Cameras that can output full RGB images directly have firmware that performs the demosaicing, but this comes with the caveat that the output is then scaled to the 8 bit range usual for digital color images.

As detailed in article **PV**, various demosaicing methods have been developed for colour cameras with the aim of reducing interpolation artifacts. However, most methods rely on assumed inter-channel correlations to improve the result quality. This makes them unsuitable for use with sensors using narrowband filtering due to the nonuniformity of the colour component responses and correlation across the range of imaged wavelengths. The same concern has been raised with interpolation methods used with multispectral colour filters (Mihoubi et al., 2017). In the case of FPI colour cameras the correlations are a function of both the FPI air gap and the transmittances of the colour filters, which complicates the issue of determining the best method for demosaicing. In principle, gradient-corrected interpolations such as the method of Malvar et al., 2004 (notably used in Matlab, see Mathworks, 2019) could be optimized for each setting of the FPI air gap

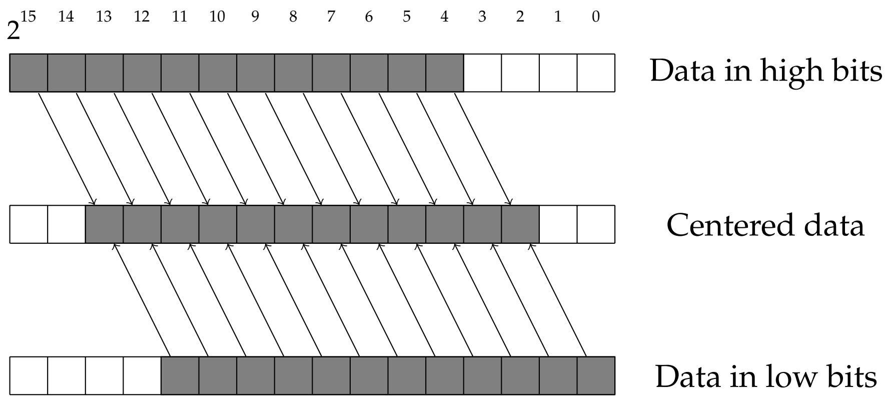


FIGURE 8 Scaling 12-bit data in a 16-bit container for interpolation using right bitshifts (top) and left bitshifts (bottom), corresponding respectively to division and multiplication by 4. Grey blocks indicate bits containing measured data. The center-packed form allows bitshifts by 2 to either direction without loss of precision.

using the original procedure for finding the best-fit coefficients for each air gap. This procedure would however require a concentrated effort to produce suitable ground truth and simulated or measured datasets for the optimization, which so far has not been attempted.

Bilinear interpolation is data-agnostic and can be implemented computationally efficiently as a 2D convolution. An additional advantage can be gained for image sensors with 12-bit ADCs, since interpolation can then be performed without loss of precision using only unsigned 16 bit integers. Given a Bayer filter mosaic image I with a set pattern, the first step is to separate it into appropriately padded colour layers. The padding should be performed first on the actual mosaic, since if it is performed on the separated layers more complicated padding methods need to be applied in order to preserve the correct sampling for each component.

Bilinear interpolation consists only of multiplications or additions totaling up to a maximum value of 4×2^{12}, and division by at most 4. Given the 4 extra bits of precision in the 16-bit container format, any 12-bit data can be scaled such that it occupies the 12 center bits (see figure 8). As all the needed coefficients are multiples of two, additional performance can be squeezed out by performing the multiplications and divisions using bit shifts. Listing 4.2 shows the Python implementation of the computation in `fpipy`. The implementation utilises the `numpy` Python library for array padding and indexing, and uses the non-padded array as an accumulator to minimize the array copies needed.

If data about the pixel format of the imaging device is present with the data, the scaling can also be automatically performed by the post processing software performing the demosaicing. If the data is eventually stored in a floating-point or other format with extra precision, the scaling can then be reversed to obtain values equal to the non-scaled procedure without loss of precision.

4.4 Radiance computation

Once full RGB information is available, pseudoradiances can be computed for each calibrated FPI transmission peak with center wavelength λ as the dot product of the RGB values and the relative peak coefficients $S^{-1}(\lambda)$ using the equation 13.

Given the exposure time and a strictly multiplicative gain value, the pseudoradiance can then be further divided with them to obtain a physically representative radiance value, giving the equation

$$L(\lambda) = \frac{R \cdot S_{\mathrm{R}}^{-1}(\lambda) + G \cdot S_{\mathrm{G}}^{-1}(\lambda) + B \cdot S_{\mathrm{B}}^{-1}(\lambda)}{V_{\mathrm{gain}} \cdot T_{\mathrm{exposure}}}. \tag{14}$$

4.5 Averaging reflectance factors

As defined in equation (5) in section 2.1, the bidirectional reflectance factor can be expressed as the ratio of radiance measurements of the surface under study and a reflectance reference surface. Given measurements of the surface and reference using the same imaging geometry and parameters, the reflectance factor for a pixel (x, y) in the image can be calculated following the definition as

$$R(x,y) = \frac{\mathrm{d}L_{\mathrm{r}}(x,y)}{\mathrm{d}L_{\mathrm{r}}^{\mathrm{id}}(x,y)}, \tag{15}$$

considering the measured values at each pixel as estimates of the pointwise radiance measurements.

As single pixels are often noisy, it is common for the hyperspectral analyst to compute an average reflectance of some number N of pixels from an image in order to smooth out noise with the assumption that the reflectance is constant across the pixels and any noise is due to the measurement. This is then usually computed by taking the mean of the previously computed pixel reflectance factors as

$$R = \frac{1}{N}\sum_{x,y} R(x,y) = \frac{1}{N}\sum_{x,y} \frac{L_{\mathrm{r}}(x,y)}{L_{\mathrm{r}}^{\mathrm{id}}(x,y)}. \tag{16}$$

However, taking a look at the definition raises questions similar to those raised by (Schaepman-Strub et al., 2006) on the confusion of definitions. If we wish to integrate R over an area in order to compute an average, and start by manipulating it to the form (omitting this time the angles)

$$\mathrm{d}L_{\mathrm{r}}^{\mathrm{id}}(x,y)R(x,y) = \mathrm{d}L_{\mathrm{r}}(x,y), \tag{17}$$

making the same assumption as before about R being constant across the area and integrating over it gives us the result

$$R = \frac{\int_A \mathrm{d}L_{\mathrm{r}}(x,y)\mathrm{d}x\mathrm{d}y}{\int_A \mathrm{d}L_{\mathrm{r}}^{\mathrm{id}}(x,y)\mathrm{d}x\mathrm{d}y}. \tag{18}$$

This results in a equation for the reflectance measured over a given area,

$$R = \frac{\sum_{x,y} L_{\mathrm{r}}(x,y)}{\sum_{x,y} L_{\mathrm{r}}^{\mathrm{id}}(x,y)}, \tag{19}$$

which is clearly a different quantity from the one in equation (16) if the reflected radiance from the ideal reflector is not constant across all the pixels. A visualisation of the difference is presented in figure 9.

This difference should have no real significance in applications where averaging is used as a noise reduction technique. However, the averaged values are also used to compare reflectances from instruments with differing pixel sizes in terms of the surface area they image, such as when comparing spectrometer measurements to imager output. In that case, the difference between the equations means that averaging should be performed at the radiance level to provide a more accurate estimate of the reflectance seen by the instrument with the larger pixel size. The author has found no mention of this discrepancy in the literature or consensus on the matter in discussions with colleagues (including some at metrology institutes), and as such the matter should be further investigated.

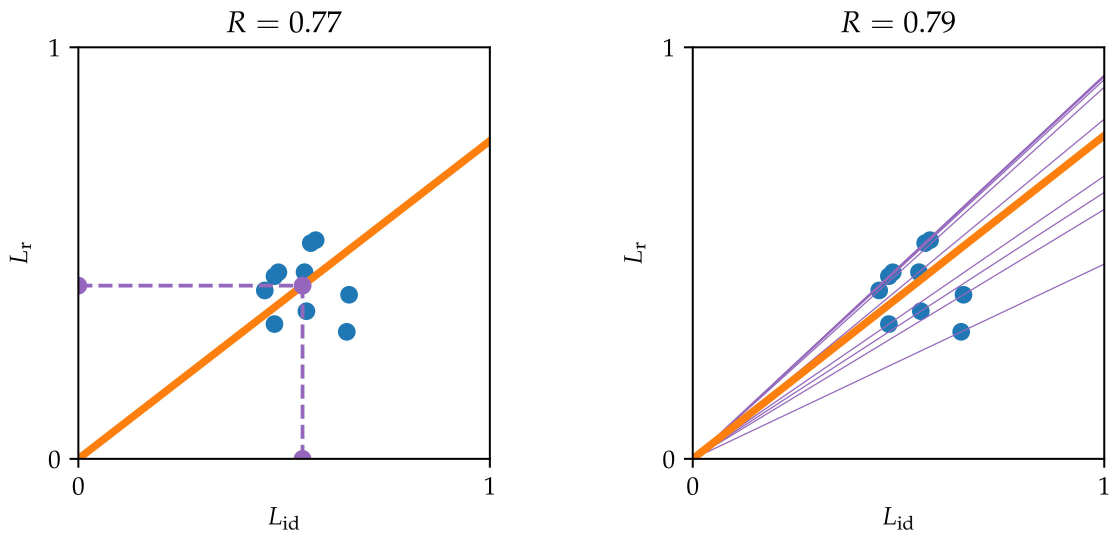


FIGURE 9 Different reflectance factors R computed from the same simulated pairs of values of L_r and L_{id} between 0 and 1 in arbitrary units. Left: The reflectance factor of the collection of points is the slope of the line (orange) passing through the mean of the simulated values (purple). Right: The mean reflectance is the average of the slopes of the lines passing through the simulated points (purple). A line with the average slope (orange) is plotted for comparison. The values given by equations (16) and (19) differ for these points by 0.02.

```
def demosaic_12bit_centered(cfa, masks):
        rgb = cfa * masks
        padded = np.pad(
            res,
            [(0, 0), (1, 1), (1, 1)],
            mode='reflect'
        )

        rgb = rgb << 2

        rgb[0, ::] += padded[0,  :-2, 1:-1] << 1
        rgb[0, ::] += padded[0,   2:, 1:-1] << 1
        rgb[0, ::] += padded[0, 1:-1,  :-2] << 1
        rgb[0, ::] += padded[0, 1:-1,   2:] << 1
        rgb[0, ::] += padded[0,  :-2,  :-2]
        rgb[0, ::] += padded[0,   2:,  :-2]
        rgb[0, ::] += padded[0,  :-2,   2:]
        rgb[0, ::] += padded[0,   2:,   2:]

        rgb[1, ::] += padded[1,  :-2, 1:-1]
        rgb[1, ::] += padded[1,   2:, 1:-1]
        rgb[1, ::] += padded[1, 1:-1,  :-2]
        rgb[1, ::] += padded[1, 1:-1,   2:]

        rgb[2, ::] += padded[2,  :-2, 1:-1] << 1
        rgb[2, ::] += padded[2,   2:, 1:-1] << 1
        rgb[2, ::] += padded[2, 1:-1,  :-2] << 1
        rgb[2, ::] += padded[2, 1:-1,   2:] << 1
        rgb[2, ::] += padded[2,  :-2,  :-2]
        rgb[2, ::] += padded[2,   2:,  :-2]
        rgb[2, ::] += padded[2,  :-2,   2:]
        rgb[2, ::] += padded[2,   2:,   2:]

        return rgb >> 2
```

LISTING 4.2 Demosaicing of 12-bit data within a 16-bit format. The function takes as parameters a colour filter mosaic `cfa` with size $x \times y$ from the camera and a $3 \times x \times y$ array of bit masks corresponding to the Bayer pattern of the mosaic.

5 CONCLUDING REMARKS AND FUTURE STUDY

The development of miniaturized hyperspectral imagers combining Fabry–Perot interferometers and machine vision cameras promises to enable many new applications for hyperspectral imaging, as they are both cheaper and more versatile than most existing solutions. The development of open source post-processing and analysis software libraries in a general purpose language such as Python marks a departure from many standard industry tools which are either proprietary or use more specialised programming languages such as Matlab or IDL (used by the ENVI software). This enables interoperability between hyperspectral imaging software and the much larger ecosystem of open source software tools for machine learning and application development, which together with the hardware developments should make hyperspectral imaging much more attractive as a tool to both industry and academia. The raw data format used as input by the `fpipy` library defines a minimal set of data that needs to be gathered from the calibration and camera to effectively compute radiance data from FPI colour cameras for any given FPI and camera combination, answering research question 1. The software implementations presented in **PIV** conclusively answer research question 2 by allowing the complete workflow from camera to analysis to be performed in Python. The author hopes that the developed software libraries also find use in future studies and applications.

The FPI imagers are still a new technology, and as such are not as well characterized or understood as other sensor types when it comes to their radiometric and spectral performance. Recent studies at national metrology institutes (Hakala et al., 2018; Pekkala et al., 2019) have found issues with insufficient factory calibrations of devices using the FPI technology, leading to spectral inaccuracies and signal leakage from extraneous interference peaks of the FPI passing through to the sensor. While the usability of the FPI colour cameras has been demonstrated in **PI**, both in terms of being able to capture data and utilizing it in a machine learning application, the various issues with the signal quality do limit the imagers in comparison to many other imager types with more established calibration procedures. While research question 3 can be considered solved for applications that do not require absolute radiometric calibration, further research should be done to improve the signal quality and reliability of the imagers and enable their use in more demanding applications. Besides radiometric calibration, improvements in signal quality could also be obtained through the development of better demosaicing methods or by utilizing the programmable camera control to tune the exposure time based on the FPI air gap to increase light capture at wavelengths with little light. The use of the raw mosaic data in applications by itself also presents an interesting area of study, as bypassing the need for radiance calculation would decrease the computational requirements considerably and enable much faster analysis of the data.

YHTEENVETO (SUMMARY IN FINNISH)

Väitöskirjatyössä tutkittiin Fabry–Perot-interferometrien ja värikameroiden avulla toteutettuja hyperspektrikameroita ja kuvantamiseen vaadittavia laskennallisia menetelmiä. Kuvannuslaitteistossa interferometria käytetään aallonpituussuotimena, jonka avulla kameran kennolle voidaan päästää kerrallaan joukko kapeita aallonpituuskaistoja. Kuvattavia aallonpituuksia voidaan vaihtaa muuttamalla interferometrin peilien etäisyyttä. Mikäli kennolle saapuva valo koostuu korkeintaan kolmesta aallonpituuskaistasta, eri kaistojen tuottamat signaalit voidaan erotella käyttäen kuluttajavärikameroissa yleisesti käytettyä värisuodatinmatriisia ja tarkoitukseen kehitetyn kalibraatiomenetelmän avulla laitteistolle määritettyjä kertoimia. Tähän tekniikkaan perustuvia VTT:n valmistamia laitteita on hyödynnetty esimerkiksi lennokki- ja satelliittisovelluksissa. Väitöskirjan ensimmäisessä julkaisussa osoitettiin että lennokkiin liitetyn kameran kuvista voidaan arvioida koneoppimismenetelmillä turpeen kosteutta.

Hyperspektrikuvien ohjelmallinen käsittely vaatii paljon kirjanpitoa, jotta eri kameroista saatavien kuvien vertailu ja datan käyttö analyysissä on mahdollista. Ensimmäisessä julkaisussa esitetyn analyysin toteuttaminen olemassaolevilla työkaluilla Matlab-ympäristössä osoitti tarpeen kehittää parempia työkaluja tutkimustyön avuksi. Tämän kehitystyön tuloksena kehitetyt ohjelmistokirjastot Matlab- ja Python-ympäristöille on esitelty toisessa ja kolmannessa julkaisussa. Mainitun signaalierottelun suorittaminen useille kuville vaatii erikoistettuja algoritmeja, jotta laskenta saadaan suoritettua tehokkaasti ja ilman ylimääräistä laskennasta johtuvaa epätarkkuutta. Viidennessä julkaisussa osoitettiin simulaatioilla, että värisuodatinkuvien käsittelyyn kehitetyt interpolointialgoritmit voivat aiheuttaa aallonpituudesta riippuvaa epätarkkuutta, mikäli niitä käytetään aallonpituussuotimia käyttävän kuvannusjärjestelmän datankäsittelyssä. Väitöskirjassa esitetään tekijän kehittämät ratkaisut kuvien mustan tason korjaukseen ja interpolointiin, jotka on julkaistu vapaasti saataville osana tekijän kehittämää `fpipy`-ohjelmistokirjastoa. Väitöskirja esittelee myös yleisemmin hyperspektrikuvantamiseen liittyviä laskennallisia ongelmia liittyen spektridatan keskiarvoistamiseen ja vertailukelpoisuuteen.

Kameroihin liittyä laskennallisten menetelmien tutkimus on tehty Jyväskylän yliopiston spektrikuvantamislaboratoriolla hyödyntäen VTT:n toimittamia prototyyppilaitteita, jotka käyttävät erikoisvalmisteisia interferometreja ja kaupallisesti saatavilla olevia konenäkökameroita. Kameroiden pieni koko ja tekniikalla saavutettava kuvausnopeus mahdollistavat niiden käytön uusissa hyperspektrikuvantamisen sovelluksissa. Neljäs artikkeli esitteleelaboratoriolla Python-ympäristöön kehitettyä ohjelmistokokonaisuutta, jonka datankäsittely perustuu tutkimuksen aikana kehitettyihin menetelmiin.

BIBLIOGRAPHY

Bayer, B. E. 1976. United States Patent: 3971065 - Color imaging array. 3971065. July 1976.

Drusch, M., Del Bello, U., Carlier, S., Colin, O., Fernandez, V., Gascon, F., Hoersch, B., Isola, C., Laberinti, P., Martimort, P., Meygret, A., Spoto, F., Sy, O., Marchese, F., and Bargellini, P. 2012. Sentinel-2: ESA's Optical High-Resolution Mission for GMES Operational Services. Remote Sensing of Environment. The Sentinel Missions - New Opportunities for Science 120 (May 2012), 25–36. ISSN: 0034-4257. DOI: 10.1016/j.rse.2011.11.026.

EMVA 2019. GenICam (Generic Interface for Cameras). ⟨URL: https://www.emva.org/standards-technology/genicam/⟩ [visited on 2019-10-24].

Fossum, E. R. and Hondongwa, D. B. 2014. A Review of the Pinned Photodiode for CCD and CMOS Image Sensors. IEEE Journal of the Electron Devices Society 2 (3) (May 2014), 33–43. ISSN: 2168-6734. DOI: 10.1109/JEDS.2014.2306412.

Gao, L. and Wang, L. V. 2016. A review of snapshot multidimensional optical imaging: Measuring photon tags in parallel. Physics Reports 616 (Feb. 2016), 1–37. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2015.12.004.

Gat, N. 2000. Imaging spectroscopy using tunable filters: a review. In Wavelet Applications VII. Vol. 4056. International Society for Optics and Photonics, Apr. 2000, 50–64. DOI: 10.1117/12.381686.

Hakala, T., Markelin, L., Honkavaara, E., Scott, B., Theocharous, T., Nevalainen, O., Näsi, R., Suomalainen, J., Viljanen, N., Greenwell, C., and Fox, N. 2018. Direct Reflectance Measurements from Drones: Sensor Absolute Radiometric Calibration and System Tests for Forest Reflectance Characterization. Sensors 18 (5) (May 2018), 1417. DOI: 10.3390/s18051417.

Hoyer, S. and Hamman, J. 2017. xarray: N-D labeled arrays and datasets in Python. Journal of Open Research Software 5 (1). DOI: 10.5334/jors.148.

Lapray, P.-J., Wang, X., Thomas, J.-B., and Gouton, P. 2014. Multispectral Filter Arrays: Recent Advances and Practical Implementation. Sensors 14 (11) (Nov. 2014), 21626–21659. DOI: 10.3390/s141121626.

Malvar, H. S., He, L.-W., Cutler, R., and Way, O. M. 2004. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In International Conference of Acoustic, Speech and Signal Processing. Institute of Electrical and Electronics Engineers, Inc., May 2004, 5–8.

Mannila, R., Näsilä, A., Viherkanto, K., Holmlund, C., Näkki, I., and Saari, H. 2013. Spectral imager based on Fabry-Perot interferometer for Aalto-1 nanosatellite. In Imaging Spectrometry XVIII. Vol. 8870. International Society for Optics and Photonics, Sept. 2013, 887002. DOI: 10.1117/12.2023299.

Manolakis, D. 2016. Hyperspectral imaging remote sensing : physics, sensors, and algorithms. Cambridge, United Kingdom: Cambridge University Press. ISBN: 978-1-107-08366-0.

Mathworks 2019. Convert Bayer pattern encoded image to truecolor image - MATLAB demosaic. ⟨URL: https://se.mathworks.com/help/images/ref/demosaic.html⟩ [visited on 2019-10-31].

Mihoubi, S., Mathon, B., Thomas, J.-B., Losson, O., and Macaire, L. 2017. Illumination-robust multispectral demosaicing. In 2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA). ISSN: 2154-512X. Nov. 2017, 1–6. DOI: 10.1109/IPTA.2017.8310135.

Minařík, R., Langhammer, J., and Hanuš, J. 2019. Radiometric and Atmospheric Corrections of Multispectral μMCA Camera for UAV Spectroscopy. Remote Sensing 11 (20) (Jan. 2019), 2428. DOI: 10.3390/rs11202428.

Nakamura, J. 2006. Image sensors and signal processing for digital still cameras. Boca Raton, FL: Taylor & Francis. ISBN: 0-8493-3545-0.

Newton, I. 1704. Opticks: or, A treatise of the reflections, refractions, inflections & of light. Royal Society.

Nicodemus, F. E., Richmond, J. C., Hsia, J. J., Ginsberg, I. W., and Limperis, T. 1977. Geometrical considerations and nomenclature for reflectance. Tech. rep. NBS MONO 160. Gaithersburg, MD: National Bureau of Standards, NBS MONO 160. DOI: 10.6028/NBS.MONO.160.

Pekkala, O., Pulli, T., Kokka, A., and Ikonen, E. 2019. Setup for characterising the spectral responsivity of Fabry–Perot-interferometer-based hyperspectral cameras. Metrologia 56 (6) (Oct. 2019), 065005. ISSN: 0026-1394. DOI: 10.1088/1681-7575/ab3fd1.

Perot, A. and Fabry, C. 1899. On the Application of Interference Phenomena to the Solution of Various Problems of Spectroscopy and Metrology. The Astrophysical Journal 9 (Feb. 1899), 87. ISSN: 0004-637X, 1538-4357. DOI: 10.1086/140557.

Saari, H., Aallos, V.-V., Akujärvi, A., Antila, T., Holmlund, C., Kantojärvi, U., Mäkynen, J., and Ollila, J. 2009. Novel miniaturized hyperspectral sensor for UAV and space applications. In Sensors, Systems, and Next-Generation Satellites XIII. Vol. 7474. International Society for Optics and Photonics, 74741M.

Saari, H., Pölönen, I., Salo, H., Honkavaara, E., Hakala, T., Holmlund, C., Mäkynen, J., Mannila, R., Antila, T., and Akujärvi, A. 2013. Miniaturized hyperspectral imager calibration and UAV flight campaigns. In Proc. SPIE. Vol. 8889. International Society for Optics and Photonics, Oct. 2013, 88891O. DOI: 10.1117/12.2028972.

Schaepman-Strub, G., Schaepman, M., Painter, T., Dangel, S., and Martonchik, J. 2006. Reflectance quantities in optical remote sensing—definitions and case studies. Remote Sensing of Environment 103 (1) (July 2006), 27–42. ISSN: 00344257. DOI: 10.1016/j.rse.2006.03.002.

Vane, G., Green, R. O., Chrien, T. G., Enmark, H. T., Hansen, E. G., and Porter, W. M. 1993. The airborne visible/infrared imaging spectrometer (AVIRIS). Remote sensing of environment 44 (2-3), 127–143.

ORIGINAL PAPERS

PI

REMOTE SENSING OF 3-D GEOMETRY AND SURFACE MOISTURE OF A PEAT PRODUCTION AREA USING HYPERSPECTRAL FRAME CAMERAS IN VISIBLE TO SHORT-WAVE INFRARED SPECTRAL RANGES ONBOARD A SMALL UNMANNED AIRBORNE VEHICLE (UAV)

by

Eija Honkavaara, Matti Eskelinen, Ilkka Pölönen, Heikki Saari, Harri Ojanen, Rami Mannila, Christer Holmlund, Teemu Hakala, Paula Litkey, Tomi Rosnell, Niko Viljanen, Merja Pulkkanen 2016

IEEE Transactions on Geoscience and Remote Sensing, 54 (9). DOI: 10.1109/TGRS.2016.2565471

Remote Sensing of 3-D Geometry and Surface Moisture of a Peat Production Area Using Hyperspectral Frame Cameras in Visible to Short-Wave Infrared Spectral Ranges Onboard a Small Unmanned Airborne Vehicle (UAV)

Eija Honkavaara, Matti A. Eskelinen, Ilkka Pölönen, Heikki Saari, Harri Ojanen, Rami Mannila, Christer Holmlund, Teemu Hakala, Paula Litkey, Tomi Rosnell, Niko Viljanen, and Merja Pulkkanen

***Abstract*—Miniaturized hyperspectral imaging sensors are becoming available to small unmanned airborne vehicle (UAV) platforms. Imaging concepts based on frame format offer an attractive alternative to conventional hyperspectral pushbroom scanners because they enable enhanced processing and interpretation potential by allowing for acquisition of the 3-D geometry of the object and multiple object views together with the hyperspectral reflectance signatures. The objective of this investigation was to study the performance of novel visible and near-infrared (VNIR) and short-wave infrared (SWIR) hyperspectral frame cameras based on a tunable Fabry–Pérot interferometer (FPI) in measuring a 3-D digital surface model and the surface moisture of a peat production area. UAV image blocks were captured with ground sample distances (GSDs) of 15, 9.5, and 2.5 cm with the SWIR, VNIR, and consumer RGB cameras, respectively. Georeferencing showed consistent behavior, with accuracy levels better than GSD for the FPI cameras. The best accuracy in moisture estimation was obtained when using the reflectance difference of the SWIR band at 1246 nm and of the VNIR band at 859 nm, which gave a root mean square error (rmse) of 5.21 pp (pp is the mass fraction in percentage points) and a normalized rmse of 7.61%. The results are encouraging, indicating that UAV-based remote sensing could significantly improve the efficiency and environmental safety aspects of peat production.**

Manuscript received October 21, 2015; revised February 2, 2016; accepted March 31, 2016. This work was supported by the Finnish Funding Agency for Technology and Innovation Tekes through the HSI-Stereo project and the Measurement, Monitoring and Environmental Efficiency Assessment research program coordinated by Cleen Ltd.

E. Honkavaara, T. Hakala, P. Litkey, T. Rosnell, and N. Viljanen are with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, 02430 Masala, Finland (e-mail: eija.honkavaara@nls.fi; teemu.hakala@nls.fi; paula.litkey@nls.fi; tomi.rosnell@nls.fi; niko.viljanen@nls.fi).

M. A. Eskelinen and I. Pölönen are with the Department of Mathematical Information Technology, University of Jyväskylä, 40014 Jyväskylä, Finland (e-mail: matti.a.eskelinen@student.jyu.fi; ilkka.polonen@jyu.fi).

H. Saari, H. Ojanen, R. Mannila, and C. Holmlund are with VTT Microelectronics, 02044 Espoo, Finland (e-mail: heikki.saari@vtt.fi; harri.ojanen@vtt.fi; rami.mannila@vtt.fi; christer.holmlund@vtt.fi).

M. Pulkkanen is with Vapo Oy Clean Waters, 40100 Jyväskylä, Finland (e-mail: merja.pulkkanen@vapo.fi).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TGRS.2016.2565471

***Index Terms*—Calibration, geographic information system, geometry, image classification, radiometry, remote sensing, remotely piloted aircraft, spectroscopy, stereo vision.**

I. Introduction

REMOTE sensing using small unmanned airborne vehicles (UAVs) is a rapidly emerging technology. UAV-based remote sensing offers possibilities for cost-efficient data collection with desired spatial and temporal resolutions, which opens up completely new remote sensing applications and new possibilities to perform scientific studies in our environment [1], [2].

An appropriate sensor is a fundamental component of a UAV remote sensing system. The first operational, civil, and lightweight UAV imaging systems typically used commercial video cameras or still cameras operating in three wide-bandwidth bands in red, green, blue (RGB) and/or near-infrared spectral regions [3]–[5]. Miniaturized hyperspectral sensors have become available to UAV platforms, offering enhanced possibilities for remote sensing applications. Hyperspectral remote sensing employs tens to hundreds of contiguous bands to accurately reconstruct the spectral signature of the target of interest [6]. The first miniaturized sensors operated in the visible to near-infrared spectral (VNIR) range extending to approximately 400–1000 nm. Several pushbroom-type hyperspectral sensors have recently been implemented in UAVs [7]–[11]. Researchers have also implemented point-based spectrometers in UAVs [12], [13]. Lately, novel hyperspectral cameras operating in a frame format principle have entered the market, such as the Rikola Hyperspectral Camera (http://www.rikola.fi), Cubert UHD 185-Firefly (http://cubert-gmbh.de/), or the IMEC SM5X5 (http://www2.imec.be). The frame sensors can be further classified based on the imaging principle as ones capturing all bands simultaneously (snapshot imaging) or as those capturing unregistered bands [14]. The methods for capturing images with unregistered bands include the time-sequential principle or multiple cameras.

When considering different sensing principles, the advantages of the frame imaging approach include the possibility to collect image blocks with stereoscopic multiple object views

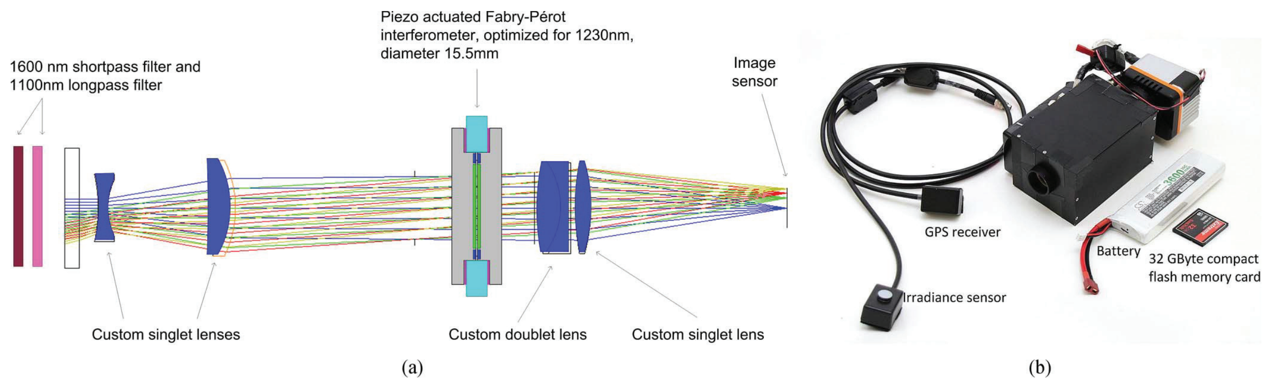

1600 nm shortpass filter and
1100nm longpass filter
Piezo actuated Fabry-Pérot interferometer, optimized for 1230nm, diameter 15.5mm
Image sensor
Custom singlet lenses
Custom doublet lens
Custom singlet lens
(a)
GPS receiver
Irradiance sensor
Battery
32 GByte compact flash memory card
(b)

Fig. 1. (a) Principle of the optical system of FPI SWIR camera prototype 2014. (b) Components of the FPI SWIR imaging system.

and the geometric and radiometric constraints provided by the rigid rectangular image geometry and multiple overlapping images [14], [15]. These can be seen as important advantages in comparison to classical hyperspectral imaging methods based on the pushbroom scanning technology or on point-based spectral measurements. This is valuable in particular for UAV applications, which typically operate under dynamic, vibrating, and turbulent conditions. Furthermore, in many applications, 3-D information is a significant feature parallel to spectral information.

This study investigates novel hyperspectral imaging technology based on a variable air gap Fabry–Pérot interferometer (FPI). The FPI technology makes it possible to manufacture a lightweight frame format hyperspectral imager operating on the time-sequential principle. The first prototypes of the FPI-based cameras were operating in the VNIR spectral range [16]–[18], and recently, a short-wave infrared (SWIR) region prototype operating in the spectral range of 1100–1600 nm was presented [19]. The FPI technology has also become commercially available in the VNIR range (http://www.rikola.fi). In the UAV operation, the camera is operated using photogrammetric principles, capturing image blocks with stereoscopic overlaps. It is crucial to emphasize the data postprocessing steps that are required to transform these huge amounts of images into products that allow the objects' geometric and spectral characteristics to be interpreted on a quantitative geometric and radiometric basis. Recently, the usability of stereoscopic frame images has improved to a new level due to the development of the structure-from-motion-based image orientation techniques [20] and the dense digital matching technologies generating accurate 3-D point clouds and digital surface models (DSMs) [21]–[26]. These modern computer vision and photogrammetric techniques are capable of providing high-quality 3-D geometric data in a highly automated way.

Surface moisture is one of the key parameters in various environmental and hydrological applications, such as agricultural water management and catchment management [27]–[29]. This work investigates the potential of FPI sensors in estimating surface moisture of a peat production area aiming at improving the efficiency and safety aspects of peat production. When harvesting peat for energy production, peat surface moisture is a critical parameter [30]. Peat is used in energy production mainly in countries where large mires can be found, such as Finland, Sweden, Russia, and Ireland. Before using peat in a burner, it is first ground from the mire surface. When the peat grind is dry enough, it is harvested in large stacks to wait for transportation to the heating plant. Peat should be suitably dry to improve combustion at the heating plant, while avoiding spontaneous combustion in the stack. Currently, the moisture measurements are carried out by collecting samples of peat and measuring the wet and dry weight of the samples, lasting for at least 24 h. With the UAV remote-sensing-based method, the speed and efficiency of peat moisture measurement could be remarkably improved. Previous investigation showed that peat spectra had several features particularly in the SWIR region supporting classification of moisture and humification levels of peat [31]. Thermal imaging is also a potential technology for surface moisture estimation. Laboratory measurements have indicated good potential of this technology, but previous experimental results with a thermal camera from a manned aircraft platform showed weak correlation of surface moisture and temperature [30].

The objective of this investigation was to study the performance of novel FPI-based VNIR and SWIR hyperspectral frame cameras in generating a DSM and measuring surface moisture of a peat production area. The expectation was that the SWIR range data are more suitable for moisture estimation than the VNIR data, but it was of interest to compare both spectral ranges. We depict the FPI camera technology in Section II. We describe the test setup used for the empirical investigation in Section III, present the empirical results in Section IV, and discuss them in more detail in Section V.

II. FPI Spectral Camera Technology

A. *Principle of FPI-Based Spectral Imager*

The hyperspectral camera developed at the VTT Technical Research Centre of Finland (VTT) [16]–[19] is based on the use of multiple orders of a variable air-gap FPI. When the FPI is placed in front of the sensor, the wavelength of the light passing the FPI is a function of the interferometer air gap (see Fig. 1). By changing the air gap, it is possible to acquire a new set of wavelengths. The final spectral response is dependent on the light passing the FPI and the spectral characteristics of the detector. The spectral bands can be selected according to the requirements of the remote sensing task. In various implementations, three-color [15]–[18] or single-color sensors have been used (http://www.rikola.fi). The number of transmission peaks passing the FPI is one to three; thus, exposure with a single gap width provides one to three different spectral bands, when a three-color sensor is used [16]–[18]. With a single-color sensor, one spectral band is obtained for each air-gap value. The first FPI cameras operated in the VNIR range, and recently, Mannila *et al.* [19] presented the first implementation of the FPI technology in the SWIR range. The first photogrammetric and remote sensing data analyses with the FPI hyperspectral imaging technology have shown that it has excellent potential in remote sensing [15], [32]–[35].

During data collection, a predefined sequence of air-gap values is applied to capture the full spectral range. The hyperspectral data cube is thus formed in the time-sequential imaging principle. When using this technology on a moving platform, each band in the data cube exposed to a different air-gap value has a slightly different position and orientation, which has to be taken into account in the postprocessing phase. During the flight, the integration time, gain, and FPI air-gap information are stored. FPI cameras are equipped with a Global Positioning System (GPS) receiver that records the exact time of the beginning of each data cube; furthermore, the sensor electronics output the synchronization pulse of each exposure (which have not been utilized thus far). An irradiance sensor based on the Intersil ISL29004 photodetector with a spectral sensitivity range of 400–1000 nm is integrated in the camera to measure the irradiance during each exposure. The sensor is not calibrated; thus, relative broadband irradiance intensity values are obtained [15], [36]. The dark signal is collected before the flight. All data are applied to the images after the flight in the postprocessing phase, as described in Section III-C–E.

B. *Cameras Used in This Investigation*

The FPI camera prototype 2012b belonging to the Finnish Geospatial Research Institute was used to capture VNIR images [15], [18]. It is equipped with custom optics having a focal length of 10.9 mm and an f-number of 2.8. The camera has a CMOSIS CMV4000 RGB image sensor with an electronic shutter. The time difference between adjacent exposures is 0.075 s, giving a time difference between the first and last exposures in a data cube with 24 bands of 1.8 s. The sensor is used in a twice binned mode, providing an image size of 1024 × 648 pixels with a pixel size of 11 μm. The field of view (FOV) is ±18° in the flight direction, ±27° in the cross-flight direction, and ±31° at the format corner. The entire camera system weighs less than 700 g.

The SWIR range spectral imager consists of the commercial indium gallium arsenide (InGaAs) camera—the Xenics Bobcat-1.7-320, the imaging optics, the FPI module, control electronics, a battery, a GPS sensor, and an irradiance sensor (see Fig. 1) [19]. The Xenics Bobcat-1.7-320 is an uncooled InGaAs camera, with a spectral band of 0.9–1.7 μm and 320 × 256 pixels and a pixel size of 20 × 20 μm. The FPI, optics, and electronics are designed and built at VTT. The focal length of the optics is 12.2 mm, and the f-number is 3.2; the FOV is ±13° in the flight direction, ±15.5° in the cross-flight direction,

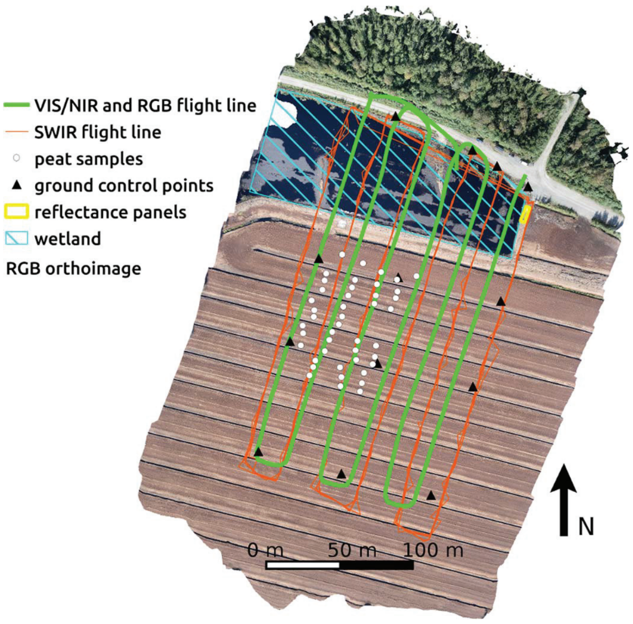


Fig. 2. Flight lines, ground control points (GCPs), and distribution of peat samples.

and ±20° at the format corner. The time between adjacent exposures is 10 ms plus exposure time; capturing single data cube with 32 bands and using 2-ms exposure time takes 0.384 s. The mass of the spectral imager unit is approximately 1200 g.

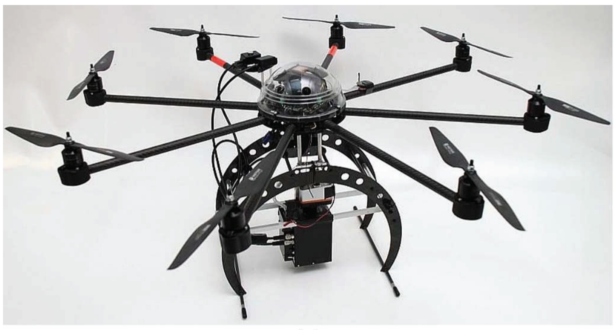
III. Materials and Methods

A. *Test Area and Flight Campaign*

Test flights were carried out in Okssuo in southern Finland (60°49′24.534″, 23°56′12.325″) on 11 September 2014. The study site is the peat production area of Vapo Oy, having a flat topography with parallel ditches and covered by spectrally homogeneous peat (see Fig. 2). The size of the area of interest was about 150 m × 150 m. Weather conditions were sunny, clear, and windless.

Image blocks with six image strips (see Fig. 2) were collected using three different cameras: the FPI VNIR camera prototype 2012b, the new FPI SWIR camera prototype, and a commercial RGB camera, the Samsung NX300. Samsung NX300 has a 23.5 × 15.7 mm CMOS RGB sensor with 20.3 megapixels and a 16-mm lens; it was used to collect high-spatial-resolution stereoscopic data for comparison and reference purposes.

For the FPI SWIR camera, we used an 8-rotor UAV, based on the MikroKopter autopilot and the Droidworx AD-8 extended frame with a 1.5-kg payload capacity. The camera was rigidly mounted to the landing gear of the UAV [see Fig. 3(a)]. The Samsung NX300 and FPI VNIR cameras were simultaneously operated using a hexacopter with a Tarot 960 foldable frame with Tarot 5008 (340 KV) brushless electric motors having a 4-kg payload capacity. The autopilot was Pixhawk equipped with Arducopter 3.15 firmware. Both cameras were rigidly mounted to the payload rails of the hexacopter [see Fig. 3(b)]. In practical operation, it is recommended to install the cameras on stabilized mount to compensate for impacts of the platform


(a)

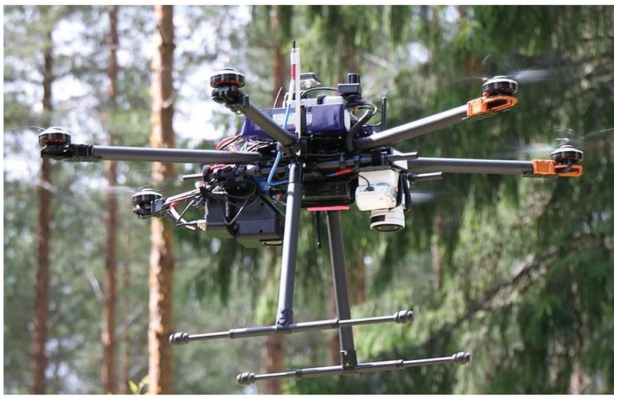

(b)

Fig. 3. (a) Lightweight SWIR camera installed in an octocopter UAV. (b) VNIR and Samsung NX300 cameras installed in a hexacopter UAV.

TABLE I
DETAILS OF THE IMAGE BLOCKS. f: FLIGHT DIRECTION; cf: CROSS-FLIGHT DIRECTION; FOV: FIELD OF VIEW

Sensor	SWIR	VNIR	RGB
Spectral sensitivity (nm)	1100-1600	500-900	R, G, B
Flight height (m)	89	94	94
GSD (m)	0.15	0.095	0.025
Exposure time (ms)	2	15	0.5
Footprint f; cf (m)	38; 47	67; 106	90; 135
Overlap f, cf (%)	77; 43	76; 78	92; 83
FOV f, cf (º)	±13°;±15.5°	±20; ±29	±26; ±36
Flight Speed (m/s)	4	4	4
Number of images	157	117	235
Time (UTC +0)	11:48-12:00	11:19-11:28	11:19-11:28
Sun elevation; azimuth (º)	31; 208	32; 199	32; 199

vibrations and fast movements in the image quality. The use of stabilized mount was not possible in this investigation due to experimental setups.

The photogrammetric block setup was designed for the SWIR camera at a flying height of 90 m above ground level. The resulting ground sample distance (GSD) was 15 cm for the SWIR camera, 9.5 cm for the VNIR camera, and 2.5 cm for the RGB camera. In the case of the SWIR camera, the size of the image footprint was 38 m $\times$ 47 m, the forward overlap was 77%, and the side overlap was 43% on average. For the FPI VNIR camera and the RGB camera, the overlaps were larger (see Table I).

The spectral settings of the FPI VNIR and SWIR cameras were selected so that the spectral range was covered quite evenly (see Table II). A total of 32 spectral bands were collected by the FPI SWIR camera in the spectral range 1100–1600 nm with the full width of half maximum (FWHM) ranging from 20 to 30 nm and with an exposure time of 2 ms. With the VNIR camera, 38 bands were collected on a spectral range of 500–900 nm having an FWHM of 11–31 nm and an exposure time of 15 ms. The long exposure time was used to obtain a good dynamic range in the relatively dark peat surface (reflectance < 0.3). The bright reflectance panels with the nominal reflectance of 0.5 were saturated with this setting; hence, it is not suitable for applications having reflectance values brighter than 0.5, for example, vegetation remote sensing. The long exposure time could also cause image quality reduction due to motion blur. However, the movement of the platform was less than the GSD during the exposure, and the weather conditions were excellent, i.e., low winds and no turbulence; thus, no significant motion blur was expected. Visual inspection of images did not show noticeable motion blur.

B. Ground Reference

We deployed 13 GCPs targeted with circular targets with a diameter of 30 cm (see Fig. 2). Their coordinates were measured using the virtual reference station real-time kinematic GPS (VRS-GPS) method with accuracy levels [root mean square error (rmse)] of approximately 3 cm in X and Y and 4 cm in Z coordinates [37].

For reflectance transformation purposes, reflectance panels of size 1 m $\times$ 1 m and with nominal reflectivity of 0.03, 0.1, and 0.5 were installed in the area. Materials of the panels were carefully selected to provide uniform reflectance properties and low anisotropy; black and dark gray panels were made of carpet, whereas the brightest panel was a painted panel [38]. The reference reflectance values were measured in a laboratory with an estimated accuracy level of 2%–5% using the FIGIFIGO goniospectrometer [39].

Altogether, 44 peat samples of size $0.05\ \mathrm{m} \times 0.05\ \mathrm{m} \times 0.03$ m were taken and measured for reflectance and moisture in the laboratory. Spatial locations of samples (XYZ coordinates) were measured with VRS-GPS. Moisture content (MC [%]) of the samples was determined based on wet weight (w_{wet}) and dry weight (w_{dry}), i.e.,

$$MC = 100\, w_{\mathrm{wet}}/(w_{\mathrm{dry}} + w_{\mathrm{wet}})[\%]. \qquad (1)$$

Sample moisture varied from 50.0% to 78.4%, and the average moisture was 67.4%.

C. Data Processing

Hundreds of small-format UAV images were collected to cover the area of interest. Rigorous processing was required to derive quantitative information from the imagery. The processing of FPI camera images is similar to any frame format camera images; the major difference is the processing of the nonoverlapping spectral bands. The FPI data processing line for MC estimation contained the following steps:

1) applying laboratory calibration corrections to the images;
2) determination of the geometric imaging model, including interior and exterior orientations of the images;

TABLE II
SPECTRAL SETTINGS OF THE FPI VNIR AND SWIR CAMERAS. L0: CENTRAL WAVELENGTH; FWHM: FULL WIDTH AT HALF MAXIMUM; DT: TIME DIFFERENCE TO THE START OF THE DATA CUBE; DS: COMPUTATIONAL SPATIAL DISTANCE TO THE START OF THE DATA CUBE

VNIR L0 (nm):	507.6, 509.5, 514.5, 520.8, 529.0, 537.4, 545.8, 554.4, 562.7, 574.2, 583.6, 590.4, 598.8, 605.7, 617.5, 630.7, 644.2, 657.2, 670.1, 677.8, 691.1, 698.4, 705.3, 711.1, 717.9, 731.3, 738.5, 751.5, 763.7, 778.5, 794.0, 806.3, 819.7, 833.7, 845.8, 859.1, 872.8, 885.6
VNIR FWHM (nm):	11.2, 13.6, 19.4, 21.8, 22.6, 20.7, 22.0, 22.2, 22.1, 21.6, 18.0, 19.8, 22.7, 27.8, 29.3, 29.9, 26.9, 30.3, 28.5, 27.8, 30.7, 28.3, 25.4, 26.6, 27.5, 28.2, 27.4, 27.5, 30.5, 29.5, 25.9, 27.3, 29.9, 28.0, 28.9, 32.0, 30.8, 27.9
VNIR dt to start of the data cube (s):	0.9, 0.975, 1.05, 1.125, 1.2, 1.275, 1.35, 1.425, 1.5, 1.575, 1.65, 1.725, 1.8, 0, 0.075, 0.15, 0.225, 0.3, 0.375, 0.45, 0.525, 0.6, 0.675, 0.75, 0.825, 0.9, 0.975, 1.05, 1.125, 1.2, 1.275, 1.35, 1.425, 1.5, 1.575, 1.65, 1.725, 1.8
VNIR ds to start of the data cube (m):	3.6, 3.9, 4.2, 4.5, 4.8, 5.1, 5.4, 5.7, 6, 6.3, 6.6, 6.9, 7.2, 0, 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3, 3.3, 3.6, 3.9, 4.2, 4.5, 4.8, 5.1, 5.4, 5.7, 6, 6.3, 6.6, 6.9, 7.2
SWIR L0 (nm):	1154.1, 1168.6, 1183.7, 1199.2, 1214.3, 1228.2, 1245.7, 1261.2, 1281.7, 1298.6, 1312.9, 1330.7, 1347.2, 1363.2, 1378.7, 1396.7, 1408.1, 1426.3, 1438.5, 1452.6, 1467.0, 1479.4, 1491.8, 1503.8, 1516.7, 1529.3, 1541.6, 1553.3, 1565.5, 1575.5, 1581.9, 1578.3
SWIR FWHM (nm):	27.0, 27.0, 26.5, 26.1, 26.7, 27.0, 26.4, 26.3, 26.2, 26.5, 26.6, 25.8, 25.8, 25.6, 26.5, 27.4, 26.9, 28.2, 27.2, 27.1, 28.6, 27.7, 27.9, 27.2, 28.8, 28.5, 28.9, 29.8, 30.4, 27.5, 20.5, 20.1
SWIR dt to start of the data cube (s):	0, 0.012, 0.024, 0.036, 0.048, 0.06, 0.072, 0.084, 0.096, 0.108, 0.12, 0.132, 0.144, 0.156, 0.168, 0.18, 0.192, 0.204, 0.216, 0.228, 0.24, 0.252, 0.264, 0.276, 0.288, 0.3, 0.312, 0.324, 0.336, 0.348, 0.36, 0.372
SWIR ds to start of the data cube (m):	0, 0.05, 0.10, 0.14, 0.19, 0.24, 0.29, 0.34, 0.38, 0.43, 0.48, 0.53, 0.58, 0.62, 0.67, 0.72, 0.77, 0.82, 0.86, 0.91, 0.96, 1.01, 1.06, 1.10, 1.15, 1.2, 1.25, 1.30, 1.34, 1.39, 1.44, 1.49

3) using dense image matching to create a DSM;
4) determination of a radiometric imaging model to transform the digital numbers (DNs) data into reflectance;
5) calculating the hyperspectral image mosaics;
6) estimating surface moisture.

The processing of FPI VNIR images is a well-developed process [15], whereas the FPI SWIR sensor is a new prototype, and the processing required additional development. In the following sections, the geometric (2, 3) and radiometric (1, 4, 5) processing steps and estimation process (6) used in this investigation are described.

D. Geometric Processing

Geometric processing determines the image orientations and creates point clouds and DSMs. Because the orientation of each band of the FPI data cube (typically 20–40 bands) would be computationally heavy, we have developed an approach that determines the orientations of selected reference bands and uses a less demanding band-matching procedure for the rest of the bands [15]. The reference bands are selected so that the temporal range of the images is covered as uniformly as possible. Different subsets of data were processed as follows.

1) Single FPI VNIR camera channel 16 (central wavelength of the band: L0 = 631 nm; spatial difference to the beginning of the data cube: ds = 0.6 m) was processed to study the geometric performance of single FPI camera band data (118 images).
2) The RGB images and three FPI VNIR camera bands 4 (L0 = 521 nm; ds = 4.5 m), 12 (L0 = 590 nm; ds = 6.9 m), and 16 (L0 = 631 nm; ds = 0.6 m) were simultaneously processed to provide the most accurate orientations for the FPI reference bands (589 images).
3) Five bands of the FPI SWIR camera were simultaneously processed to provide orientations for the SWIR data cubes and SWIR DSM (983 images). The bands were 3 (L0 = 1184 nm; ds = 0.1 m), 8 (L0 = 1261 nm; ds = 0.3 m), 11 (L0 = 1313 nm; ds = 0.5 m), 24 (L0 = 1504 nm; ds = 1.1 m), and 28 (L0 = 1553 nm; ds = 1.3 m).
4) The RGB images were used to create an accurate DSM that was used as reference for other data sets (235 images).

Agisoft PhotoScan Professional commercial software (AgiSoft LLC, St. Petersburg, Russia) was used to determine the image orientations and to generate dense point clouds over the object area. Its excellent performance has been validated in previous studies [24], [35]. PhotoScan performs photo-based 3-D reconstruction using feature detection and dense matching. In each data set, the images were automatically oriented without *a priori* orientation information. The GPS flight trajectory information could also be used to provide the approximate orientations, which could make the processing faster. In the orientation processing, the PhotoScan quality setting was set to "high"; settings for the number of key points per image were 40 000 and those for the final number of tie points per image were 1000; an automated lens calibration was simultaneously performed. According to our experiences, these settings are suitable for the FPI images to provide accurate results and reasonable processing time. An automatic outlier removal was performed using the tools of the software on the basis of the reprojection error (10% of points with the largest errors were removed) and reconstruction uncertainty (10% of points with the largest errors were removed). Finally, some points were manually removed from a sparse cloud, particularly points up in the air or underground. This processing provided the orientation of the images and sparse point clouds in the internal coordinate system of the software.

The object reference coordinate system information was used to transform the image orientations into the desired coordinate system. We used different control data configurations to evaluate if the system has a consistent geometric performance and to study optimal georeferencing configurations: 1) 13 GCPs; 2) positions for images measured by the autopilot's GPS and no GCPs; 3) GPS and one GCP close to the takeoff location; 4) GPS and four GCPs in block corners; 5) GPS and five GCPs (four in block corners and one in the center of the block); 6) four GCPs in block corners; and 7) five GCPs (four in block corners and one in the center of the block). In practical operation, the configurations with a minimum number of GCPs are the most efficient. The standard deviation settings for the GCPs were $\sigma_{\text{GCP\_XYZ}} = 0.001$ m, and for the GPS coordinates, we used a standard deviation of $\sigma_{\text{GPS\_XYZ}} = 3$ m. The projection accuracy was set to 0.1 pixels, and tie point accuracy was set to 4 pixels. The outputs of the final self-calibrating block adjustment were the camera calibrations, and the image exterior orientations and sparse point clouds in the ETRS TM35FIN coordinate system.

In the dense point cloud generation process, for VNIR and SWIR camera images, the full-resolution images were used, and for the RGB point cloud generation, four times downscaled images were used. Depth filtering was used to filter out outliers in the point clouds. For the SWIR data, the PhotoScan quality setting "mild" was used, performing the least filtering. For the VNIR data and for the RGB data, the setting "moderate" was used, assuming a flatter object and to eliminate more height points.

A band-matching procedure was used for the bands that were not included in the orientation processing. Band matching was carried out using a feature-based matching algorithm, and an affine transformation was used to map the bands to the reference bands. In the previous investigations, the accuracy of this approach has been shown to be on the level of a pixel in flat areas [15]. In this matching, the spatial difference to the reference band (derived from ds; Table II) is an important factor impacting the quality of band matching. For the VNIR camera, we used the reference band for each major spectral color range corresponding to the bands that were oriented by PhotoScan: band 4 (L0 = 521 nm; ds = 4.5 m), band 16 (L0 = 631 nm; ds = 0.6 m), and band 29 (L0 = 764 nm; ds = 4.5 m). For the SWIR camera, we used the same five reference bands (3, 8, 11, 24, and 28) that were oriented in the PhotoScan processing and matched each unoriented band to the temporally closest reference band. For the noisiest bands in the atmospheric absorption region (bands 15–21; L0 1379–1467 nm), we interpolated the orientations from the orientation trajectory determined photogrammetrically in the PhotoScan processing because the matching would not have provided a reliable result.

The geometric accuracy was evaluated by using independent check points and evaluating the DSMs. The 3-D point determination accuracy was assessed using the GCPs that were not included in the georeferencing and VRS-GPS coordinates of the peat sample points as check points. Height accuracy and deformations of the VNIR and SWIR DSMs were assessed by using the RGB DSM as reference. The accuracy of alignment of bands of final image mosaics was evaluated by using an image correlation technique, by matching all the bands to a reference band, and by calculating discrepancies.

E. Radiometric Modeling and Reflectance Mosaic Generation

Radiometric modeling includes the sensor corrections, the atmospheric correction, correction for the illumination changes and other nonuniformities, and the normalization of illumination and viewing-direction-related nonuniformities by utilizing the bidirectional reflectance distribution function (BRDF) correction.

The sensor corrections for the FPI images include spectral smile correction, photon response nonuniformity correction (PRNU), and dark signal correction [15], [16]. The PRNU and smile corrections were determined at the laboratory of VTT [16]. The dark signal correction is calculated using a black image collected right before the data capture. In this investigation, all these correction steps were used for the FPI VNIR camera. For the FPI SWIR camera, only the dark signal correction was used; developing laboratory calibration procedures for the prototype sensor was not possible in this investigation. As the SWIR images showed significant sensor-related nonuniformities, we developed a series of empirical corrections, as described in Section III-E1.

The reflectance transformation was carried out using the empirical line method [40] with the aid of the reflectance panels in the area. For the SWIR images, all panels with nominal reflectance of 0.03, 0.10, and 0.5 were used. For the VNIR images, the brightest panel was not used because it was saturated in most of the bands. The model was

$$DN = a_{\text{abs}} Refl + b_{\text{abs}} \tag{2}$$

where a_{abs} and b_{abs} are the parameters for the empirical line model for transforming the reflectance $(Refl)$ to DN. The transformation was calculated using the image where the panels were the closest to the image center to avoid impacts of reflectance anisotropy in the reference reflectance.

To correct for the atmospheric instability and the impacts of BRDF, a radiometric block adjustment approach was used [15]. The basic principle of the approach is to use the DNs of the radiometric tie points in the overlapping images as observations and to determine the parameters of the radiometric model indirectly via the least squares principle. The model for a DN is

$$DN = a_{\text{rel\_}j}\left(a_{\text{abs}} R_{\text{jk}}(\theta_i, \theta_r, \varphi) + b_{\text{abs}}\right) \tag{3}$$

where $R_{\text{jk}}(\theta_i, \theta_r, \varphi)$ is the bidirectional reflectance factor of the object point, k, in image j; θ_i and θ_r are the illumination and reflected light (observation) zenith angles; φ_i and φ_r are the azimuth angles, respectively; $\varphi = \varphi_r - \varphi_i$ is the relative azimuth angle; and $a_{\text{rel\_}j}$ is the relative image-wise correction parameter. The linear BRDF model by Walthall [41] was used to correct the BRDF effects. The estimated nadir reflectance (R_{nadir}) is

$$R_{\text{nadir}} = (DN/a_{\text{rel\_}j} - b_{\text{abs}}) / \left(a_{\text{abs}}\left(a'\theta_r^2 + b'\theta_r\cos\varphi + 1\right)\right) \tag{4}$$

where a′ and b′ are adjustable BRDF model parameters. Homogeneous distribution of radiometric tie points was generated, with approximately 70 tie points in each image; the DN observation of tie points was calculated in an image window of size 3 m × 3 m (see details in [15]).

In this investigation, the final image output was a reflectance mosaic. Image mosaics were resampled with a 20-cm GSD from the image block data with the aid of the image orientations and DSM and applying the radiometric model. The reflectance values were taken for each mosaic pixel from the image where the image ray had the smallest difference to the vertical direction. For the VNIR images, we used the full model (4) to calculate reflectance values. We carried out several empirical corrections for the SWIR images (see below), and because

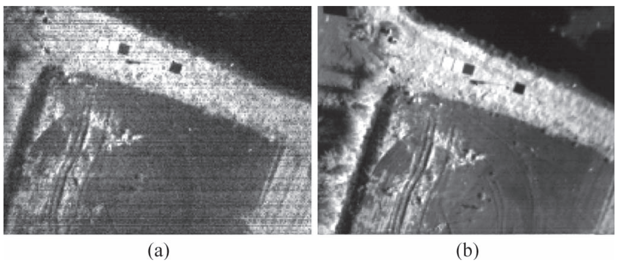


Fig. 4. Sample images of FPI SWIR bands (a) 21 and (b) 30.

of this, it was feasible to estimate only the relative parameters $(a_{\mathrm{rel\_}j})$ in the radiometric block adjustment. It was not possible to solve the BRDF parameters because the empirical corrections strongly correlated with the BRDF effects; thus, default values $a' = b' = 0$ were used. The irradiance observations could be used to calculate the $a_{\mathrm{rel\_}j}$ parameters [36], but in this study, the irradiance values were not used.

1) Empirical Radiometric Calibration of the FPI SWIR Camera: The first analysis of the spectra and image mosaics calculated of the FPI SWIR camera images without any corrections indicated that the radiometric calibration was not accurate enough. The mosaics were not homogeneous due to the missing lens falloff calibration. Second, there appeared a decrease in image intensity during the flight and negative DN values. The most probable reason for this distortion was the change in the dark signal during the flight due to sensor cooling. Furthermore, there was a relatively high level of noise in the images (see Fig. 4).

A series of empirical corrections was applied to the images to eliminate these distortions. These corrections were calculated by assuming that the target area had uniform reflectance on average in each spectral band. The wetland area in the northern part of the area was not included in these calculations. All the corrections were calculated separately for each band. To eliminate the impacts of the changes in the dark signal, a strip-wise additive dark current correction was calculated by assuming that the average reflectance of each strip should be the same. The correction was calculated based on the average DNs of each strip in comparison to the average DN of the reference strip (in this case, the first strip). A median-image-based approach was used to eliminate the lens falloff. The assumption was that the median image calculated using all images (except the wetland area) should show uniform intensity, whereas the nonuniformity of the median image indicates systematic radiometric distortions. A multiplicative pixel-wise correction coefficient $LFC(l, m)$ with respect to the central pixel of the median image $(\mathrm{med\_image}(\mathrm{row}_c, \mathrm{col}_c))$ was calculated for each pixel (l, m), i.e.,

$$LFC(l, m) = \mathrm{med\_image}(\mathrm{row}_c, \mathrm{col}_c)/\mathrm{med\_image}(l, m). \quad (5)$$

The corrected DN is

$$DN_{\mathrm{cal}}(l, m) = LFC(l, m)DN(l, m). \quad (6)$$

F. Remote Sensing of Surface Moisture

The albedo of peat is known to decrease nonlinearly on wetting, with additional changes in the shape of the spectrum particularly near the water absorption features. However, in the range of MCs measured, the changes in albedo could be expected to be approximately linear [31]. We studied the usefulness of spectral features for moisture estimation by calculating linear correlations between the features and moisture and by employing machine learning for the study of nonlinear dependence relations. The data sets used in the study of peat moisture estimation were the VNIR mosaic with BRDF and relative image-wise corrections; the SWIR mosaic with dark signal correction, median-image-based calibration, and relative image-wise corrections; and the RGB mosaic with standard PhotoScan processing scaled to range 0–1 by dividing the DNs by 255.

Single-pixel reflectance spectra were collected from the mosaics from the locations corresponding to the collected samples (20 cm $\times$ 20 cm area); tests using larger sample areas showed reduced correlation with the measured moisture. Reflectance differences $(R_i - R_j)$ and ratios (R_i/R_j) were calculated for each pair of bands of the concatenated RGB, VNIR, and SWIR spectra of each sample. Linear correlation of the individual bands, band differences, and ratios to the MC was examined using the MATLAB function corr to calculate the Pearson correlation for each set of features. A machine-learning approach based on a support vector machine (SVM) was employed for moisture estimation to take into account possible nonlinearities. A leave-one-out approach was used to estimate the performance of different data sets for the SVM machine learning model generation. The SVM was trained for each set of feature vectors and moistures corresponding to 43 samples and then used to predict the moisture of the remaining sample. The process was repeated for each sample with the individual prediction errors collected together and used to calculate the performance statistics for each data set. SVM training and prediction was performed using ν—SVR (support vector regression) with libSVM [43]. Optimum C and γ parameters for the SVM were determined for a grid of parameters in the $\log_2$ C $\times$ $\log_2$ γ space, and selecting for each data set the parameters that resulted in the lowest mean square error in the procedure.

TABLE III
STATISTICS OF GEOMETRIC PROCESSING: NUMBER OF IMAGES, TIE POINTS, AND PROJECTIONS; REPROJECTION ERROR AND NUMBER OF POINTS AND POINT DENSITY IN DENSE POINT CLOUD

Dataset	N ima.	N Tie points	N Proj.	Re-proj. Error (pix)	Dense point cloud: N points; points/m2
SWIR	983	61018	238888	0.323	3.8 e6; 45
VNIR ch 16	118	6914	37344	0.480	9.0e6; 111
RGB	235	10264	154985	0.994	13.5e6; 103
RGB + VNIR	589	14538	260353	0.893	-

IV. RESULTS

A. Geometric Performance

Statistics of the geometric processing indicated accurate results (see Table III). The reprojection errors were on the level of 5–7 μm in image coordinates for all calculations (about 0.5 pixels for the FPI VNIR, 0.3 pixels for FPI SWIR, and

TABLE IV
ERROR STATISTICS OF THE VARIOUS GEOMETRIC PROCESSING CONFIGURATIONS FOR FPI VNIR IMAGES. N GCP: NUMBER OF GCPS; GPS: YES = GPS SUPPORT USED, NO = GPS SUPPORT NOT USED; N CP: NUMBER OF CHECK POINTS

N GCP;	N CP	Mean (m)			Standard deviation (m)			RMSE (m)		
GPS		X	Y	Z	X	Y	Z	X	Y	Z
0; yes	13	-0.019	0.034	-0.765	0.171	0.217	0.081	0.166	0.211	0.769
1; yes	12	0.473	0.279	-0.436	0.297	0.344	0.376	0.552	0.432	0.566
4; yes	9	-0.011	0.001	-0.008	0.019	0.025	0.045	0.021	0.024	0.043
5; yes	8	-0.012	-0.026	-0.026	0.016	0.022	0.050	0.019	0.033	0.053
4; no	9	-0.014	0.002	0.163	0.020	0.025	0.105	0.023	0.023	0.191
5; no	8	-0.014	-0.024	-0.031	0.016	0.021	0.057	0.021	0.032	0.061

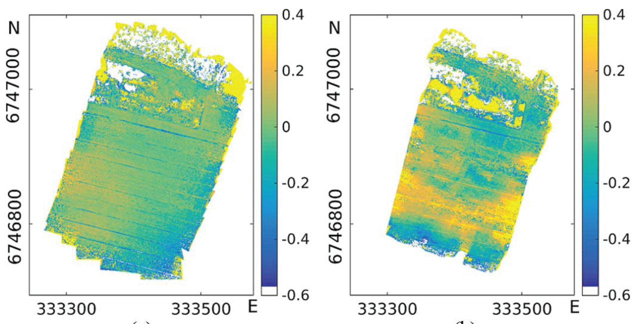


Fig. 5. Differences to the reference DSM in meters: (a) RGB—VNIR and (b) RGB—SWIR.

one pixel for RGB cameras). The point densities in dense point clouds were about 100 points/m^2 for RGB and FPI VNIR cameras and 45 points/m^2 for FPI SWIR camera. Processing of the SWIR images was less stable than processing of VNIR and RGB cameras, which was due to the poorer block structure, smaller image format, and worse image quality.

The 3-D point determination accuracy of an FPI VNIR data set processed with different georeferencing configurations was studied using the GCPs that were not included in the orientation as check points (see Table IV); the visibility of GCPs was poor in SWIR images because of the larger GSD and the noisier image quality; thus, SWIR images were not analyzed. The best rmse was on the level of 2–3 cm in the X and Y coordinates and 5–6 cm in height. This accuracy level was obtained when using four GCPs with the autopilot's GPS support or five or more GCPs without the GPS support. The case with the GPS support and no GCPs was of quite low geometric quality, with an rmse of about 0.2 m in X and Y and 0.8 m in Z.

The results of DSM accuracy assessment are shown in Fig. 5 when using all 13 GCPs in georeferencing and no GPS support; these results represent the best achievable accuracy. The results suggested that the VNIR DSM had a slight tilt (less than 20 cm) to the reference surface; the west side was higher than the east side [see Fig. 5(a)]. The SWIR DSM had more deviation from the reference surface with south–north-aligned distortion, which was less than 40 cm [see Fig. 5(b)]. In particular, the VNIR DSM was not significantly deformed in the area surrounded by GCPs. When using the peat sample points as check points, the height rmses were approximately 10–12 cm, at best, for all of the materials (see Table V). The mean errors showed a moderate negative bias of 9–11 cm, which could be due to the difference in measuring the height of the peat surface in the field and from the image. The height standard deviations were on the level of 4 cm for the RGB and FPI VNIR cameras and about 8 cm for the FPI SWIR camera. Height rmses were 2–5 cm for the GCPs, indicating that the DSM fitted very well to these points. One potential explanation for the better height accuracy results with the GCPs is the better measurement accuracy of the well-defined surface of the GCP.

Analysis of the impact of the ground control configuration in the DSM height error is presented in Figs. 6 and 7. In the case of the SWIR data set, the DSM did not show significant deformation [see Figs. 5(b) and 6(f)] when using the configurations with 5 or 13 GCPs and no GPS support; these cases provided also a good height rmse in the peat sample points, on the level of 11 cm (see Table V). In all cases with the GPS support, the SWIR DSM surface was deformed [see Fig. 6(a)–(e)], and the height rmses were large, 0.35–3 m in check points (see Table V); the most likely explanation for the poor results is the low quality of the autopilot GPS solution during the SWIR flight. For the VNIR data, the different GCP configurations provided consistent results with the previous analysis when using GCPs as reference (see Tables IV and V and Fig. 5). The DSM did not show deformations in the cases where the rmses were low [see Figs. 5(a) and 7(c), (d), and (f)]; for the cases with poor rmses, the DSMs were deformed [see Fig. 7(a), (b), and (e)].

Assessment of the quality of alignment of individual spectral bands of the mosaics showed good results. For the VNIR data, the mosaic of band 4 was used as reference for all the bands. The discrepancies in the X and Y coordinates were less than 1 pixel in 90%–99% of the matched points in bands 1–34 and in 85%–90% of the matched points in bands 35–38. For the SWIR data, mosaics of bands 3 and 28 were used as reference. In most of the bands, the discrepancies were less than 1 pixel in X and Y coordinates in 80%–99% of matched points; in bands 12–14, the discrepancies were less than 1 pixel in 70%–80% of matched points. In the bands in the atmospheric absorption region for which the orientations were interpolated (bands 15–21), a majority of discrepancies were on the level of 3 pixels and less. These results showed that the individual bands were well aligned.

TABLE V
STATISTICS OF THE DSM ASSESSMENT FOR THE VARIOUS GEOMETRIC PROCESSING CONFIGURATIONS WHEN USING THE 44 INDEPENDENT CHECK POINTS (CP) AND 13 GCPS AS THE REFERENCE. MEAN, STANDARD DEVIATION (STD), AND RMSE OF ERRORS (ERROR = REFERENCE − INTERPOLATED FROM DSM). N GCP: NUMBER OF GCPS; GPS: YES = GPS USED, NO = GPS NOT USED

Data set	N GCP; GPS	mean_{CP} (m)	std_{CP} (m)	RMSE_{CP} (m)	mean_{GCP} (m)	std_{GCP} (m)	RMSE_{GCP} (m)
RGB	13; no	-0.109	0.040	0.116	0.010	0.020	0.021
VNIR	13; no	-0.092	0.041	0.100	0.013	0.058	0.057
	4; no	0.406	0.048	0.417	-0.106	0.134	0.167
	5; no	-0.120	0.046	0.129	0.023	0.050	0.053
	0; yes	0.567	0.053	0.569	0.760	0.108	0.767
	1; yes	0.412	0.151	0.438	0.394	0.370	0.531
	4; yes	-0.140	0.049	0.148	0.009	0.054	0.053
	5; yes	-0.130	0.045	0.137	0.024	0.048	0.052
SWIR	13; no	-0.086	0.077	0.115	-0.002	0.030	0.029
	4; no	0.923	0.197	0.943	0.443	0.599	0.726
	5; no	-0.072	0.083	0.110	0.000	0.068	0.065
	0; yes	2.255	0.543	2.318	2.598	1.132	2.816
	1; yes	-0.043	0.526	0.522	0.363	1.370	1.365
	4; yes	-0.310	0.216	0.376	0.039	0.386	0.373
	5; yes	-0.403	0.198	0.448	-0.020	0.371	0.357

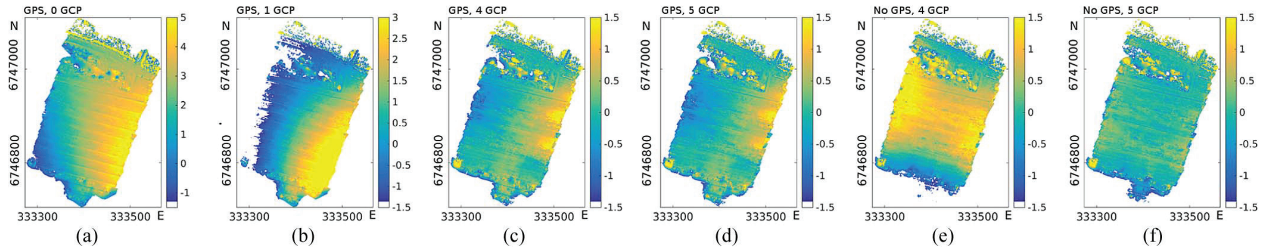


Fig. 6. Differences (in meters) between the FPI SWIR DSMs and the reference DSM (RGB—SWIR) when using different georeferencing configurations: (a) GPS; (b) GPS, 1 GCP; (c) GPS, 4 GCPs; (d) GPS, 5 GCPs; (e) no GPS, 4 GCPs; (f) no GPS, 5 GCPs.

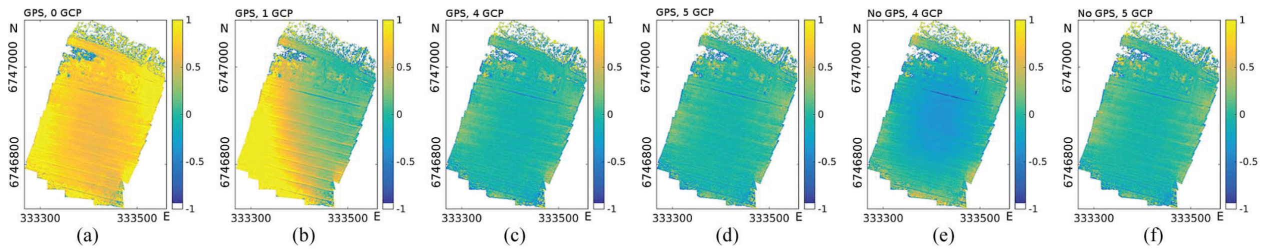


Fig. 7. Differences (in meters) between the FPI VNIR DSMs and the reference DSM (RGB—VNIR) when using different georeferencing configurations: (a) GPS; (b) GPS, 1 GCP; (c) GPS, 4 GCPs; (d) GPS, 5 GCPs; (e) no GPS, 4 GCPs; (f) no GPS, 5 GCPs.

B. Radiometric Processing

The incomplete radiometric calibration was visible in the SWIR mosaics without any radiometric corrections (see Fig. 8, left). The dark signal change was visible particularly in band 21 [see Fig. 8(a)]. The dark signal correction and the median-image-based lens falloff correction compensated for most of the mosaic nonuniformity (see Fig. 8, center). However, some part of the nonuniformity still remained. This could be due to the fact that the strip-wise dark signal correction was not accurate (most likely the dark signal changed continuously due to the temperature changes). After the radiometric block adjustments using the relative image-wise corrections ($a_{\text{rel\_}j}$), the mosaics were uniform (see Fig. 8, right) and suitable for the following remote sensing analysis. A three-band SWIR mosaic is shown in Fig. 9(a).

The BRDF correction and the relative image-wise corrections ($a_{\text{rel\_}j}$) were used when calculating the VNIR mosaics. The relative corrections were on the level of 10%. The view angle range in the FPI VNIR images is 0 to $\pm 31^\circ$ from the nadir, which is expected to cause BRDF effects in images. The hemispherical directional reflectance factor plot of an area of about 150 m $\times$ 150 m with a point interval of 10 m indicated that the peat surface was backward scattering with decreasing reflectance toward a forward scattering direction (see Fig. 10). The reflectance anisotropy was about 10% with a view angle range of 0 to $\pm 25^\circ$ and about 20% from maximum to minimum. The output mosaics of VNIR data had uniform data quality [see Fig. 9(b) and (c)].

C. Peat Spectra and Moisture Estimation

The use of spectral information in the assessment of the peat moisture is based on the fact that the more moist the peat is, the darker it is. Some fluctuations appeared in the peat spectra,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

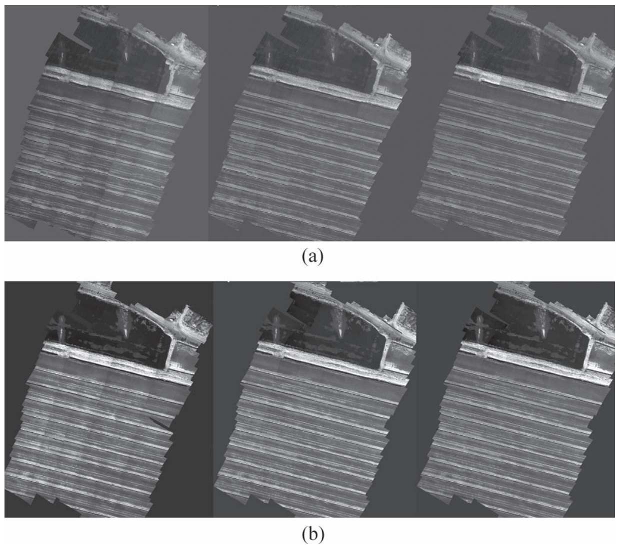


Fig. 8. Examples of reflectance mosaics with different processing options for SWIR bands (a) 21 and (b) 30. Processing versions from left: original images without corrections; dark signal correction and median image calibration; dark signal correction, median image calibration, and relative correction.

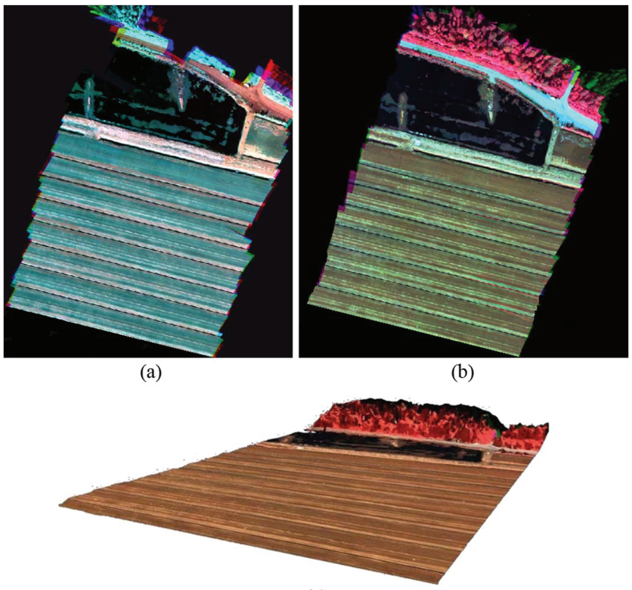


Fig. 9. Hyperspectral reflectance mosaic from (a) SWIR data (bands at 1184, 1331, and 1553 nm) and (b) VNIR data (bands at 520.80, 630.70, and 763.70 nm). (c) 3-D perspective visualization of the VNIR mosaic.

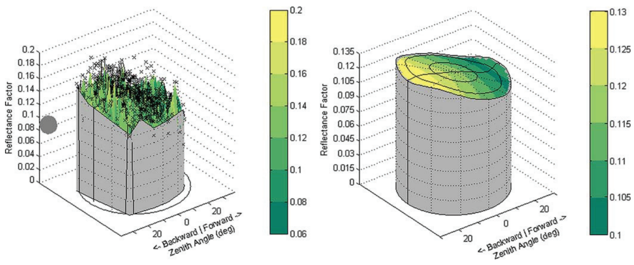


Fig. 10. Reflectance factor observations as the function of the view and illumination angles (left) and fitted BRDF surface (right) for near-infrared band L0 = 764 nm.

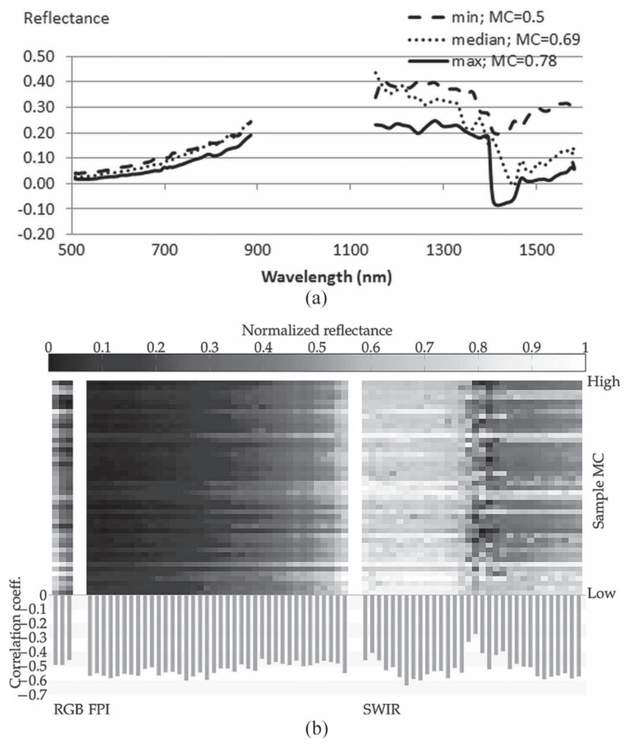


Fig. 11. (a) Spectra of peat samples with the minimum, maximum, and median MC. (b) Reflectance spectra of all the samples and the correlations of each band with the measured MC for the RGB, VNIR (FPI), and SWIR data sets. The rows and columns correspond, respectively, to different samples and bands. The spectra in each data set are normalized to fill the interval [0, 1], and the samples are arranged by their measured MC.

but the overall form of the spectra was consistent, showing an increase from green toward NIR wavelengths, a drop at the water absorption region (1350–1450 nm) and a linear increase toward longer wavelengths [see Fig. 11(a)]. Assessment of the reflectance spectra of individual bands with respect to the target moisture indicated correlations of up to 0.63 at highest for SWIR band 7 at 1246 nm; typical correlations were 0.5–0.6 with VNIR and SWIR data sets [see Fig. 11(b)]. For the RGB camera, the correlations were lower, i.e., less than 0.5. The plot shows quite consistent darkening as the MC increased. In cases where correlation is high (such as SWIR band 7), there appeared quite linear change in reflectance. In the water absorption region (1350–1450 nm), the spectra appeared noisy, and the correlations were low.

The correlation coefficients and the associated P-values of the MC and the reflectance differences $(R_i - R_j)$ and ratios (R_i/R_j) were collected into square matrices and color coded (see Fig. 12). The highest correlations (with $r = 0.66, p < 10^{-6}$) were obtained using the reflectance difference of SWIR band 7 (L0 = 1246 nm) and VNIR band 36 (L0 = 859 nm).

Table VI lists the results of the remote sensing analysis of surface moisture by the SVM. The data sets RGB, VNIR,

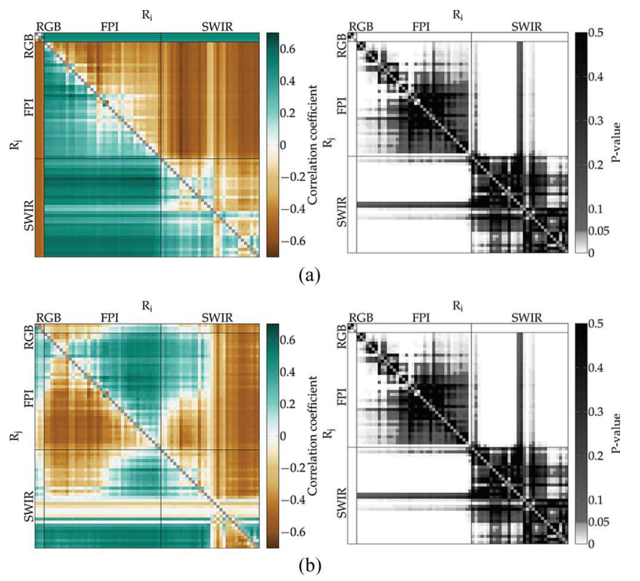


Fig. 12. Pearson correlation coefficients and associated P-values between the MC and (a) band differences $R_i - R_j$ and (b) band fractions R_i/R_j calculated using the RGB, VNIR (FPI), and SWIR data sets. Black lines were added to separate the bands corresponding to different data sets.

TABLE VI
RESULTS OF THE SVM LEAVE-ONE-OUT ANALYSIS. THE ERROR IS COMPUTED FOR EACH SAMPLE AS THE DIFFERENCE OF THE REFERENCE AND THE PREDICTED MC ($\mathrm{MC_{REF}} - \mathrm{MC_{PRED}}$; IN PERCENTAGE POINTS PP). THE STATISTICS INCLUDE MEAN ERROR, STANDARD DEVIATION, RMSE, AND NRMSE (RMSE NORMALIZED BY THE MEAN MEASURED MC)

Data	Mean (pp)	Standard deviation (pp)	RMSE (pp)	NRMSE (%)
RGB+VNIR+SWIR	-0.41	5.79	5.74	8.52
RGB	-1.38	6.31	6.39	9.48
FPI	-0.44	5.92	5.87	8.71
SWIR	-0.25	5.74	5.68	8.44
SWIR7- VNIR36	0.16	5.21	5.15	7.65

and SWIR refer to the spectral vectors for each sensor. The RGB+VNIR+SWIR data set is the set of concatenated spectral vectors from the respective data sets. The SWIR7-VNIR36 data set is the vector of reflectance differences of SWIR band 7 ($\mathrm{L0} = 1246$ nm) and VNIR band 36 ($\mathrm{L0} = 859$ nm) for each sample. The best accuracy was obtained when using the reflectance difference of SWIR band 7 and VNIR band 36, which gave the rmse of 5.21 pp (pp is the mass fraction in percentage points) and a normalized rmse (nrmse) of 7.61%. When using individual cameras, the rmses were 6.39, 5.87, and 5.68 pp, and the nrmses were 9.48%, 8.71%, and 8.44% for the RGB, VNIR, and SWIR cameras, respectively.

V. DISCUSSION

This investigation studied the performance of two novel lightweight frame format hyperspectral cameras onboard a small UAV in measuring the 3-D surface model and surface moisture of a peat production area. The VNIR FPI camera was shown to be operational in previous studies [15], [32]–[35]. The SWIR FPI camera [19] was a completely new prototype camera. The UAV system was also equipped with a high-spatial-resolution customer RGB camera. Data sets were captured over a peat production area using a flying height of 90–94 m with GSDs of 2.5, 9.5, and 15 cm for the RGB, VNIR, and SWIR cameras, respectively.

A. *Considerations on Geometric Performance*

As the FPI-based imager captures spectral data cubes using the time-sequential principle, the individual bands of the data cubes are unregistered. We used a methodology where several bands of the set of unregistered data cubes were simultaneously oriented in the self-calibrating bundle block adjustment. The success of this approach is dependent on the invariance of the matching algorithm to the spectral differences in images. The commercial Agisoft PhotoScan software was quite tolerable to the spectral differences within visible range (400–650 nm) and SWIR range (1100–1600 nm) and provided accurate integrated orientation result. The procedure is based on interest points, which are stable under viewpoint and lighting variations and their descriptors based on each points' local neighborhood. Other software programs have similar algorithms [24], and they are expected to be functional as well; however, each method needs to be confirmed. The use of several bands in the processing simultaneously made the processing more robust by providing better overlaps between the images of the block. The bands that were not included in the block adjustment were successfully registered to the oriented bands by a band-matching process. A majority of the discrepancies between the bands were one pixel or less; these results were consistent with previous studies performed with the VNIR camera in areas with flat topography [15].

We validated the geometric performance of the FPI cameras by using different sets of ground reference data, including variable configurations of GCPs and autopilots' GPS observations. The detailed analysis using the accurate targeted XYZ control points was possible for VNIR imagery. The best results were on the level of 2–3 cm in X and Y coordinates and 5–6 cm in height. The cases based only on the autopilot's GPS data showed quite low geometric accuracy, which is in line with expected quality when using navigation-grade GPS reference [44]. The best height accuracy levels were approximately 10–12 cm for all of the data sets in the independent check points located in the swamp surface. This estimate included the uncertainty of peat surface measurement and the quality of the measurement method.

The geometric performance of the SWIR data set was worse than that of other cameras, which was due to the poorer block structure (lower image and strip overlaps), potentially lower accuracy of the autopilot's GPS data, smaller image size, and lower image quality. In this investigation, the block structure was optimized for the SWIR camera, flat object, and mosaic calculation; thus, the overlap of 40%–45% was used between the image strips; as the flight trajectory was the same for the RGB and VNIR cameras that had larger FOVs, the resulting block geometry was better for those data sets. The assessment showed that the FPI-based sensors were capable of providing good DSMs. When concerning with the geometric performance of the new sensors, the potential deficiency with hyperspectral cameras in comparison to good-quality customer digital color cameras is the smaller number of pixels (poorer spatial resolution), smaller image size, and lower signal-to-noise ratio (because of measuring narrower spectral bands). Furthermore, in the SWIR range, the decreasing level of solar illumination sets additional demands on the sensor when a high signal-to-noise ratio is anticipated [42]. In the end, the propagation of errors will follow the photogrammetric theories [45]. Most importantly, the block structure (overlaps, cross-strips, GCPs) and the quality of the direct orientation observations impact the accuracy of orientations, and the level of signal-to-noise ratio impacts the quality of matching [22], [46]. To obtain high DSM quality, the overlaps within and between the strips should be at least 80% and 60%, respectively, and crossing flight strips are advantageous. In the practical operation, the recommended approach is to integrate a high-spatial-resolution RGB camera with a hyperspectral sensor, to obtain the most accurate DSM and desired spectral properties, if this is possible with the UAV platform in use.

B. *Aspects of Spectral Measurement Quality*

The new SWIR prototype had some shortcomings in the spectral measurement quality, which will be improved in the next versions of the camera. These included the missing PRNU calibration as well as the change in the dark signal during the flight. For the miniaturized sensors, temperature calibration together with a correction algorithm based on a model of the sensor behavior is likely the best approach to eliminate temperature effects, since stabilization of sensor temperature might be challenging due to the weight limitations. Several other researchers have also pointed out the importance of calibrating small hyper- and multi-spectral cameras accurately [7], [9], [14], [34], [47]. Dark signal correction based on the dark image collected before the flight appeared to be inaccurate because of the changes in the sensor temperature during the flight. A simple improvement to this procedure would be to capture the dark image before and after the campaign. More rigorous approaches would be determining the temperature impacts on the dark image in controlled conditions or integrating dark signal measurement to the data capture process. The empirical processing steps tailored for this data set were capable of compensating for these limitations and provided suitable data quality for further processes. The radiometric block adjustment approach was functional in eliminating the remaining radiometric nonuniformities in the image mosaics [15]. The accurate radiometric processing in difficult conditions will be an important step in the UAV-based hyperspectral remote sensing. In the future, the radiometric quantities in UAV remote sensing need to be carefully considered [42], [48], for example, are the images processed to reflectance or some other quantities and are the anisotropic reflectance effects compensated for. Furthermore, comprehensive studies concerning the performance of different radiometric correction approaches should be carried out.

In comparison to conventional pushbroom technology, the interesting feature of the frame format hyperspectral cameras is the possibility to collect hyperspectral image blocks with stereoscopic multiview overlaps. On the other hand, for the first-generation frame format hyperspectral cameras, the spectral resolution is still poorer (FWHM on the level of 10–30 nm), and the number of spectral bands is lower (20–40) than that of more mature pushbroom techniques, which typically provide hundreds of spectral bands with FWHMs of 2–10 nm [7]–[11]. The spectral performance of the FPI cameras is expected to improve in future systems, and the latest commercial cameras already offer improved performance (www.rikola.fi).

C. *Remote Sensing of Surface Moisture*

We evaluated the performance of UAV remote sensing with the RGB camera and the FPI-based VNIR and SWIR cameras in the measurement of surface moisture of a peat production area. The results indicated good agreement of the reflectance signatures in images with the moisture of the object. The peat moisture estimation was more accurate with the FPI SWIR camera than with the FPI VNIR or traditional RGB camera. The best accuracy was obtained when using the reflectance difference of SWIR band 7 ($L0 = 1246$ nm) and VNIR band 36 ($L0 = 859$ nm), which gave an rmse of 5.21 pp and an nrmse of 7.61%. Based on our experiences in using a series of peat samples with controlled moisture levels in a laboratory setting, higher accuracy moisture estimation from the spectra should be possible. In particular, the low reflectance and the noisiness of the UAV SWIR spectra near the main water absorption at 1400-nm feature limited the accuracy of the estimation.

Here, the training of the SVM was based on samples from the imaged mire. Due to the nonlinear effects of humification and the differences in the chemical composition of the peat [31], [49], it is not expected that an empirical model trained on one mire is generalizable to other mires. The next steps in research would be gathering a series of training material from different humification levels of peat and performing laboratory testing to separate the effects of humification and moisture on the spectra. To summarize, the major uncertainty components in our experiment included sample gathering, accuracy of positioning, accuracy of moisture measurement system, and humification level. When these factors are well controlled, we assume that the general accuracy of estimation is higher.

The presented technology allows many further improvements. We did not utilize the gathered 3-D geometry in the moisture estimation; it is expected that the topographic information would support the estimation of the surface and below-surface moisture in the peat mire. We used the most nadir approach in the analysis of the reflectance data, but in the future, the multiview information should be used more efficiently. The potential of multidirectional observations (BRDF) and a physical-radiative-transfer-based approach in estimating the MC needs further study, to develop general methods that do not require *in situ* reference. The future autonomous UAV systems are feasible for repetitive data acquisition and multitemporal analyses, which might improve the estimation accuracy levels.

Moreover, the potential of other spectral ranges, particularly multitemporal thermal imaging, should be studied [29], [30].

D. Outlook

Based on the results, the FPI hyperspectral technology follows the principles of central perspective imaging, thus enabling the utilization of latest innovations in mainstream computer vision and photogrammetric technologies in developing highly automatic and autonomous applications [1], [2], [20]–[26]. In this investigation, the geometric, radiometric, and peat surface moisture reference were based on field measurements. It will be important to develop reliable methods that do not require interactive field measurements to allow automatic and autonomous operation in the future. These methods will require accurate geometric and radiometric calibration of sensors [9], [14], [34], accurate satellite positioning methods [2], [25], automatic radiometric correction, and detailed understanding of the remote sensing task. When small differences in object characteristics, such as the MC, are of interest, the requirements for the spectral and geometric quality of the remote sensing data are high. This means that the spectral and topographic information has to be accurate and unbiased; the detailed requirements for the data quality need to be studied in the future investigations.

Peat production areas form an environmental risk factor due to the threat of self-ignition. Furthermore, efficiency of peat production could be improved if using efficient remote-sensing-based techniques in estimating optimal time for harvesting and managing the area. Our results indicated that the UAV techniques could be an efficient tool for further optimizing and monitoring of the environmental impacts of peat production. In addition to monitoring peat production areas, the presented approach can be applied for any other application requiring repetitive monitoring in relatively flat surfaces, such as in precision agriculture [15]. The technology has also been demonstrated in large areas (several km^2) (unpublished results) and in a complex 3-D environment in forest [35]; the technology is also suitable in these applications, but in complex environments, a more complicated overall solution is required.

VI. Conclusion

This paper has studied the performance of two novel lightweight FPI-based hyperspectral frame format cameras in measuring the 3-D surface model and surface moisture of a peat production area. The SWIR range camera was a new prototype, whereas the visible to near-infrared range camera was more mature technology. Moreover, a high-spatial-resolution consumer RGB camera was used. More rigid image geometry as well as the possibility for stereoscopic measurements and multiple object views are important advantages of the frame format geometry in comparison to conventional hyperspectral imaging technology based on pushbroom geometry.

The results were promising, indicating that UAV-based remote sensing could significantly improve the efficiency and the environmental safety aspects of peat production. In addition to monitoring peat production areas, the technology is functional in various remote sensing applications. As the FPI technology follows the theories of central perspective imaging, it is also well suited for developing automatic and autonomous applications utilizing the latest innovations in photogrammetry and computer vision.

References

[1] C. Zecha, J. Link, and W. Claupein, “Mobile sensor platforms: Categorisation and research applications in precision farming,” *J. Sens. Sens. Syst.*, vol. 2, no. 1, pp. 51–72, 2013.

[2] I. Colomina and P. Molina, “Unmanned aerial systems for photogrammetry and remote sensing: A review,” *ISPRS J. Photogramm. Remote Sens.*, vol. 92, pp. 79–97, Jun. 2014.

[3] E. Hunt, Jr. *et al.*, “Acquisition of NIR-green-blue digital photographs from unmanned aircraft for crop monitoring,” *Remote Sens.*, vol. 2, no. 1, pp. 290–305, 2010.

[4] C. Lelong, “Assessment of unmanned aerial vehicles imagery for quantitative monitoring of wheat crop in small plots,” *Sensors*, vol. 8, no. 5, pp. 3557–3585, 2008.

[5] G. Zhou, “Near real-time orthorectification and mosaic of small UAV video flow for time-critical event response,” *IEEE Trans. Geosci. Remote Sens.*, vol. 47, no. 3, pp. 739–747, Mar. 2009.

[6] A. Goetz, “Three decades of hyperspectral remote sensing of the Earth: A personal view,” *Remote Sens. Environ.*, vol. 113, pp. S5–S16, Sep. 2009.

[7] P. Zarco-Tejada, V. Gonzalez-Dugo, and J. Berni, “Fluorescence, temperature and narrow-band indices acquired from a UAV platform for water stress detection using a micro-hyperspectral imager and a thermal camera,” *Remote Sens. Environ.*, vol. 117, pp. 322–337, Feb. 2012.

[8] R. Hruska, J. Mitchell, M. Anderson, and N. Glenn, “Radiometric and geometric analysis of hyperspectral imagery acquired from an unmanned aerial vehicle,” *Remote Sens.*, vol. 4, no. 12, pp. 2736–2752, 2012.

[9] A. Büttner and H. Röser, “Hyperspectral remote sensing with the UAS ‘Stuttgarter Adler’—System setup, calibration and first results,” *Photogrammetrie—Fernerkundung—Geoinform.*, vol. 2014, no. 4, pp. 265–274, 2014.

[10] A. Lucieer, Z. Malenovsky, T. Veness, and L. Wallace, “HyperUAS-imaging spectroscopy from a multirotor unmanned aircraft system,” *J. Field Robot.*, vol. 31, no. 4, pp. 571–590, Jul./Aug. 2014.

[11] J. Suomalainen *et al.*, “A lightweight hyperspectral mapping system and photogrammetric processing chain for unmanned aerial vehicles,” *Remote Sens.*, vol. 6, no. 11, pp. 11 013–11 030, 2014.

[12] K. Uto, H. Seki, G. Saito, and Y. Kosugi, “Characterization of rice paddies by a UAV-mounted miniature hyperspectral sensor system,” *IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens.*, vol. 6, no. 2, pp. 851–860, Apr. 2013.

[13] A. Burkart *et al.*, “Angular dependency of hyperspectral measurements over wheat characterized by a novel UAV based goniometer,” *Remote Sens.*, vol. 7, no. 1, pp. 725–746, 2015.

[14] H. Aasen, A. Burkart, A. Bolten, and G. Bareth, “Generating 3-D hyperspectral information with lightweight UAV snapshot cameras for vegetation monitoring: From camera calibration to quality assurance,” *ISPRS J. Photogramm. Remote Sens.*, vol. 108, pp. 245–259, Oct. 2015.

[15] E. Honkavaara *et al.*, “Processing and assessment of spectrometric, stereoscopic imagery collected using a lightweight UAV spectral camera for precision agriculture,” *Remote Sens.*, vol. 5, no. 10, pp. 5006–5039, 2013.

[16] J. Mäkynen *et al.*, “Unmanned Aerial Vehicle (UAV) operated megapixel spectral camera,” in *Proc. SPIE Electro-Opt. Remote Sens., Photon. Technol., Appl. V*, 2011, vol. 8369, pp. 1–9, doi: 10.1117/12.897712.

[17] H. Saari *et al.*, “Unmanned Aerial Vehicle (UAV) operated spectral camera system for forest and agriculture applications,” in *Proc. SPIE Remote Sens. Agriculture, Ecosyst., Hydrol. XIII*, 2011, vol. 8174, pp. 1–15, doi: 10.1117/12.897585.

[18] H. Saari *et al.*, “Miniaturized hyperspectral imager calibration and UAV flight campaigns,” in *Proc. SPIE Sens., Syst., Next-Gener. Satell. XVII*, 2013, vol. 8889, pp. 1–12, doi: 10.1117/12.2028972.

[19] R. Mannila, C. Holmlund, H. Ojanen, A. Näsilä, and H. Saari, “Short-Wave Infrared (SWIR) spectral imager based on Fabry–Pérot interferometer for remote sensing,” in *Proc. SPIE Sens., Syst., Next-Gener. Satell. XVIII*, 2014, vol. 9241, pp. 1–8, doi: 10.1117/12.2067206.

[20] C. Wu, “Towards linear-time incremental structure from motion,” in *Proc. Int. Conf. 3DV*, 2013, pp. 127–134, doi: 10.1109/3DV.2013.

[21] F. Leberl *et al.*, “Point clouds: LIDAR versus 3-D vision,” *Photogramm. Eng. Remote Sens.*, vol. 76, no. 10, pp. 1123–1134, 2010.

[22] T. Rosnell and E. Honkavaara, "Point cloud generation from aerial image data acquired by a quadrocopter type micro unmanned aerial vehicle and a digital still camera," *Sensors*, vol. 12, no. 12, pp. 453–480, 2012.
[23] J. Lisein, M. Pierrot-Deseilligny, S. Bonnet, and P. Lejeune, "A photogrammetric workflow for the creation of a forest canopy height model from small unmanned aerial system imagery," *Forests*, vol. 4, no. 4, pp. 922–944, 2013.
[24] A. Eltner and D. Schneider, "Analysis of different methods for 3-D reconstruction of natural surfaces from parallel-axes UAV images," *Photogramm. Rec.*, vol. 30, no. 151, pp. 279–299, Sep. 2015.
[25] B. Jagt, A. Lucieer, L. Wallace, D. Turner, and M. Durand, "Snow depth retrieval with UAS using photogrammetric techniques," *Geosciences*, vol. 5, no. 3, pp. 264–285, 2015.
[26] S. Puliti, H. Olerka, T. Gobakken, and E. Nässet, "Inventory of small forest areas using an unmanned aerial system," *Remote Sens.*, vol. 7, no. 8, pp. 9632–9654, 2015.
[27] G. P. Petropoulos, G. Ireland, and B. Barrett, "Surface soil moisture retrievals from remote sensing: Current status, products & future trends," *Phys. Chem. Earth, A/B/C*, vol. 83/84, pp. 36–56, 2015.
[28] L. K. DeBell, K. Anderson, R. E. Brazier, N. King, and L. Jones, "Water resource management at catchment scales using lightweight UAVs: Current capabilities and future perspectives," *J. Unmanned Veh. Syst.*, vol. 3, pp. 1–24, 2015, doi: 10.1139/juvs-2015-0026.
[29] D. J. Luscombe, K. Anderson, N. Gatis, E. Grand-Clement, and R. E. Brazier, "Using airborne thermal imaging data to measure near-surface hydrology in upland ecosystems," *Hydrol. Process.*, vol. 29, no. 6, pp. 1656–1668, Mar. 2015, doi: 10.1002/hyp.10285.
[30] M. Tervo, E. Kiukaanniemi, and T. Kauppinen, "Applications of aerial thermography in peat production in Finland," in *Proc. SPIE Thermosense XV, Int. Conf. Thermal Sens. Imaging Diagnostic Appl.*, Apr. 6, 1993, vol. 101, pp. 1–9, doi: 10.1117/12.141959.
[31] J. McMorrow, A. Al-Roichdi, M. Evans, and M. Cutler, "The effect of moisture content and humification on the hyperspectral reflectance of peat," in *Proc. RSPSoc, Scales Dyn. Observ. Environ.*, Nottingham, U.K., 2003, pp. 1–11.
[32] J. Kaivosoja *et al.*, "A case study of a precision fertilizer application task generation for wheat based on classified hyperspectral data from UAV combined with farm history data," in *Proc. SPIE Remote Sens. Agriculture, Ecosyst., Hydrol. XV*, Dresden, Germany, Sep. 23–26, 2013, pp. 1–9.
[33] I. Pölönen, H. Saari, J. Kaivosoja, E. Honkavaara, and L. Pesonen, "Hyperspectral imaging based biomass and nitrogen content estimations from light-weight UAV," in *Proc. SPIE Remote Sens. Agriculture, Ecosyst., Hydrol. XV*, 2013, vol. 8887, pp. 1–9, doi: 10.1117/12.2028624.
[34] G. Bareth *et al.*, "Low-weight and UAV-based hyperspectral full-frame cameras for monitoring crops: Spectral comparison with portable spectroradiometer measurements," *Photogrammetrie—Fernerkundung—Geoinform.*, vol. 2015, no. 1, pp. 69–79, 2015.
[35] R. Näsi *et al.*, "Using UAV-based photogrammetry and hyperspectral imaging for mapping bark beetle damage at tree-level," *Remote Sens.*, vol. 7, no. 11, pp. 15 467–15 493, 2015.
[36] T. Hakala *et al.*, "Spectral imaging from UAVs under varying illumination conditions," *Int. Archives Photogramm. Remote Sens. Spatial Inf. Sci.*, vol. XL-1/W2, pp. 189–194, 2013.
[37] P. Häkli, "Practical test on accuracy and usability of virtual reference station method in finland," in *FIG Working Week*. Athens, Greece: The Olympic Spirit in Surveying, May 22–27, 2004.
[38] E. Honkavaara, L. Markelin, T. Hakala, and J. Peltoniemi, "The metrology of directional, spectral reflectance factor measurements based on area format imaging by UAVs," *Photogrammetrie—Fernerkundung—Geoinform.*, vol. 2014, no. 3, pp. 175–188, Jun. 2014.
[39] J. I. Peltoniemi *et al.*, "Technical notes: A detailed study for the provision of measurement uncertainty and traceability for goniospectrometers," *J. Quant. Spectrosc. Radiat. Transf.*, vol, 146, pp. 376–390, Oct. 2014.
[40] G. Smith and E. Milton, "The use of the empirical line method to calibrate remotely sensed data to reflectance," *Int. J. Remote Sens.*, vol. 20, no. 13, pp. 2653–2662, 1999.
[41] C. Walthall, J. Norman, J. Welles, G. Campbell, and B. Blad, "Simple equation to approximate the bidirectional reflectance from vegetative canopies and bare soil surfaces," *Appl. Opt.*, vol. 24, no. 3, pp. 383–387, 1985.
[42] J. Schott, *Remote Sensing*. New York, NY, USA: Oxford Univ. Press, 2007.
[43] C. Chang and C. Lin, "LIBSVM," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, pp. 1–27, Apr. 2011.
[44] D. Turner, A. Lucieer, and C. Watson, "An automated technique for generating georectified mosaics from ultra-high resolution Unmanned Aerial Vehicle (UAV) imagery, based on Structure from Motion (SfM) point clouds," *Remote Sens.*, vol. 4, no. 5, pp. 1392–1410, 2012.
[45] K. Kraus and P. Waldhausl, *Photogrammetry*. Bonn, Germany: Dummler, 1993.
[46] E. Honkavaara, L. Markelin, T. Rosnell, and K. Nurminen, "Influence of solar elevation in radiometric and geometric performance of multispectral photogrammetry," *ISPRS J. Photogramm. Remote Sens.*, vol. 67, no. 1, pp. 13–26, Jan. 2012.
[47] J. Kelcey and A. Lucieer, "Sensor correction of a 6-band multispectral imaging sensor for UAV remote sensing," *Remote Sens.*, vol. 4, no. 5, pp. 1462–1493, 2012.
[48] G. Schaepman-Strub, M. Schaepman, T. Painter, S. Dangel, and J. Martonchik, "Reflectance quantities in optical remote sensing—Definitions and case studies," *Remote Sens. Environ.*, vol. 103, no. 1, pp. 27–42, 2006.
[49] D. J. M. Hayes, M. H. B. Hayes, and J. J. Leahy, "Analysis of the ligno-cellulosic components of peat samples with development of near infrared spectroscopy models for rapid quantitative predictions," *Fuel*, vol. 150, pp. 261–268, Jun. 2015.

Eija Honkavaara received the D.Sc. degree in technology in photogrammetry from the Helsinki University of Technology, Espoo, Finland, in 2008.

She is currently a Research Manager and a Research Group Leader with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, Masala, Finland. Her research interests include photogrammetry, unmanned aerial vehicles, radiometric and geometric calibration, computer vision, and hyperspectral environmental monitoring applications.

Matti A. Eskelinen received the M.Sc. degree in theoretical physics from the University of Jyväskylä (JYU), Jyväskylä, Finland, in 2015. He is currently working toward the Ph.D. degree in computational hyperspectral analysis with the Department of Mathematical Information Technology, JYU.

His research interests include hyperspectral image analysis, inverse problems, machine learning, and geometric algebra.

Mr. Eskelinen is currently the Vice Chairman of the Jyväskylä Physical Society.

Ilkka Pölönen received the Ph.D. degree in information technology from the University of Jyväskylä, Jyväskylä, Finland.

He is the Head of the Spectral Imaging Laboratory with the Department of Mathematical Information Technology, University of Jyväskylä. His research interests include data analysis, mathematical modeling, and numerical simulations.

Heikki Saari received the D.Sc. degree in technology in technical physics from Helsinki University of Technology, Espoo, Finland, in 1996.

He is a Principal Scientist with the Microelectronic Systems Group, VTT Technical Research Centre of Finland, Espoo. He is the author or coauthor of more than 100 publications and 12 patents. His current research interests include micro-opto-electro-mechanical system spectrometers and Fabry–Pérot interferometer-based hyperspectral imagers in unmanned aerial vehicle remote sensing, medical, space, and industrial applications.

Harri Ojanen received the Ph.D. degree in mathematics from Rutgers University, New Brunswick, NJ, USA, in 1999.

He is currently with the Microspectrometers Group, VTT Technical Research Centre of Finland, Espoo, Finland. His research interests include the application of miniature framing imaging spectrometer systems based on Fabry–Pérot interferometers into environmental monitoring and healthcare.

Rami Mannila received the Lic.Sc. degree in physics from the University of Helsinki, Helsinki, Finland, in 2009.

He is a Senior Scientist with the VTT Technical Research Centre of Finland, Espoo, Finland. His research interests include microspectrometers, unmanned aerial vehicles, hyperspectral imagers, and spectrometer-based environmental monitoring applications.

Christer Holmlund received the Master's degree in applied electronics and the Lic.Tech. degree in measurement technology from Helsinki University of Technology, Espoo, Finland, in 1983 and 1990, respectively.

He is a Senior Scientist with the VTT Technical Research Centre of Finland, Espoo. His research interests include electronics and firmware design of microspectrometers and hyperspectral imagers.

Teemu Hakala received the M.Sc. degree in technology in electronics from Helsinki University of Technology, Espoo, Finland, in 2009. He is currently working toward the Ph.D. degree with Aalto University, Espoo.

He is a Research Scientist with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, Masala, Finland. His research interests include unmanned aerial vehicle sensor technology, radiometric measurements, hyperspectral environmental monitoring, hyperspectral LiDAR, and radar technology.

Paula Litkey received the M.Sc. and Lic.Tech. degrees in electrical engineering from Helsinki University of Technology, Espoo, Finland, in 1998 and 2004, respectively.

She is a Senior Research Scientist with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, Masala, Finland. Her research interests include point cloud applications, surface models, and change detection.

Tomi Rosnell received the M.Sc. degree in technology in photogrammetry from the Helsinki University of Technology, Espoo, Finland, in 2010.

He is a Research Scientist with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, Masala, Finland. His current research interests include indoor and outdoor close-range photogrammetry, three-dimensional modeling, and unmanned aerial vehicles.

Niko Viljanen received the B.Sc. degree in technology in geodesy and photogrammetry from Aalto University, Espoo, Finland, in 2016, where he is currently working toward the M.Sc. degree in technology in photogrammetry and remote sensing.

He is a Research Assistant with the Department of Remote Sensing and Photogrammetry, Finnish Geospatial Research Institute, Masala, Finland. His research interests include photogrammetry, unmanned aerial vehicles, geometric calibration, and hyperspectral environmental monitoring applications.

Merja Pulkkanen received the Ph.D. degree in aquatic sciences from the University of Jyväskylä, Jyväskylä, Finland, in 2013.

She is an Environmental Data Modeler with Vapo Oy Clean Waters, Jyväskylä, a startup business that provides expert, design, and construction services for the treatment of natural waters.

PII

SOFTWARE FRAMEWORK FOR HYPERSPECTRAL DATA EXPLORATION AND PROCESSING IN MATLAB

by

Matti Eskelinen 2017

ISPRS SPEC3D 2017 : Frontiers in Spectral imaging and 3D Technologies for Geospatial Solutions (pp. 47-50). International archives of the photogrammetry, remote sensing and spatial information sciences, Volume XLII-3/W3. International Society for Photogrammetry and Remote Sensing. DOI: 10.5194/isprs-archives-XLII-3-W3-47-2017

SOFTWARE FRAMEWORK FOR HYPERSPECTRAL DATA EXPLORATION AND PROCESSING IN MATLAB

Matti A. Eskelinen

University of Jyväskylä, Faculty of Information Technology, Jyväskylä, Finland - matti.a.eskelinen@student.jyu.fi

Commission III, WG III/4

KEY WORDS: Software, Hyperspectral Imaging, Data Exploration, MATLAB

ABSTRACT:

This paper presents a user introduction and a general overview of the MATLAB software package `hsicube` developed by the author for simplifying the data manipulation and visualization tasks often encountered in hyperspectral analysis work, and the design principles and software development methods used by the author. The framework implements methods for slicing, masking, visualization and application of existing functions to hyperspectral data cubes without the need to use explicit indexing or reshaping, as well as enabling expressive syntax for combining these operations on the command line for highly efficient data analysis workflows. It also includes utilities for interfacing with existing file reader scripts for easy access to files using the framework. The `hsicube` framework is released as open source to promote the free use and peer review of the code and enable collaborative development.

1. INTRODUCTION

Hyperspectral imaging data presents a challenge for data exploration due to it's high dimensionality, size and very application-specific features of interest. Programming algorithms and processing pipelines for analysis and visualization of hyperspectral data requires the programmer to keep track of multiple variables besides the data, such as wavelength information and regions of interest. Care must be taken when translating between data exploration and batch processing workflows to ensure correct results, and again when visualizing results of the batch processing to prevent erroneous presentations. Automation of the steps of such workflows — while highly desirable — is often a complex programming task due to the management of many datasets of different dimensions along with their metadata. Consider for example the following workflow:

1. Read a hyperspectral cube and its metadata from disk,
2. Extract data for region(s) of interest,
3. Select data in a specific wavelength region(s),
4. Compare select spectra visually with reference data,
5. Apply a model on the selected data and collect the results,
6. Visualize the results overlaid on the original dataset.

Steps 1 and 5 are cases that existing MATLAB libraries such as `ENVIreader/writer` (Totir and Howat, 2010) and `Hyperspectral ToolBox` (Gerg, 2016) are built to tackle, while the rest of the steps are usually constructed piece-by-piece using the basic MATLAB functionality. However, since MATLAB does not provide any smart datatypes for combining metadata with multidimensional arrays, manual bookkeeping is necessary in many of the steps if one wishes to keep the metadata synchronized with the operations done to the array. Such bookkeeping is a major source of errors and a large time expenditure for the programmer, and easily results in repetetive code if not properly abstracted. This in turn makes working inside a REPL (read-eval-print-loop) such as the MATLAB command line or iPython prohibitively cumbersome, which is detrimental to rapid prototyping of new algorithms.

Object-based datatypes that encapsulate both the data and metadata along with methods to operate on the data are a common abstraction for problems of this kind in many languages. For example, open source solutions like *Spectral Python* (Boggs, 2016) or *xarray* (Hoyer and Hamman, 2017) exist for manipulating hyperspectral or general multidimensional data in Python, but no similar datatype implementations exist for MATLAB.

This paper showcases a framework for simplifying hyperspectral data analysis workflows in a similar way using MATLAB. The framework package is mostly feature complete for the author's current workflow and a development version has already been used for the hyperspectral analysis in (Salmi et al., 2017). General design principles of the framework are detailed in section 2, with the internal implementation along with the main properties of the class listed in section 3. The implemented methods for simplifying the steps of the example workflow are presented in section 4. Section 5 contains a note about the testing methodology used for ensuring code quality, and section 6 has notes on acquiring and using the code.

2. GENERAL DESIGN

The guiding principle in the access to the internal object properties is that of minimal privilege, meaning that the programmer using the data type is purposely unable to access the object in a way that could de-synchronize the metadata and the data (at least without considerable effort). This is a design choice of the author to favor data integrity over malleability, and in contrast with the approach taken by many libraries which often allow most unsafe operations, but recommend not using them. However, the author feels that recommendations rarely deter programmers looking to

Table 1. Data type object properties

Property	Description	Example content
Data	Data array	`ones(10,10,100)`
Files	File(s) of origin	`{'data.hdr'}`
Quantity	Quantity of the data	`'Reflectance'`
WavelengthUnit	Unit of the wavelengths	`'nm'`
Wavelength	Vector of wavelengths	`[400 ... 500]`
FWHM	Vector of FWHMs	`[3, 4, 3, ..., 3]`
History	History of operations	`{'Read from file'}`
Version	Class version	`'0.8.0'`

do the easy thing instead of the right thing, and as such prefers the implemented technical solution.

Following from this, all the object properties are accessible read only, and can only be assigned new values during after object construction through the object various object methods that validate the input for the specific purpose. In addition, methods that manipulate the data but cannot generate meaningful default metadata for the results, require the user to supply valid metadata instead of generating non-meaningful defaults.

There has also been some effort to standardize the format of class method arguments to improve the user experience. As a notable divergence from the usual MATLAB syntax, the class methods use the syntax `[x,y]` for indexing horizontal and vertical pixel coordinates instead of the usual MATLAB syntax of `[y,x]`. This makes it easier to relate indices to the MATLAB image visualizations.

3. INTERNALS

The main content of the software package is a MATLAB object class named `Cube` which contains as properties the hyperspectral data and relevant metadata and provenance for the data. The properties implemented in the current version are listed in table 1. The properties are read-only after the creation of the object, and can only be changed using the object methods to ensure validity and synchronization during operations.

The object class is implemented as a MATLAB value class, which means that each object method returns a copy of the object instead of a reference (along with any other return values). This makes the state of the object easier to reason about, but relies on the MATLAB memory management to optimize memory use. The internal object methods do however try to minimize the creation of temporary extra copies of the data where possible.

The argument parsing for the Cube class constructor (see 4.1) is implemented flexibly using the class CubeArgs, which is subclassed from the MATLAB InputParser class. Due to the amount of development effort required to implement this kind of functionality in MATLAB, the "Name, value" syntax is currently only implemented for the constructor method.

Some utility functions, which are not specific to, but used by the Cube class are separated into a separate namespace in the static class Utils. ENVI file format reading functionality is similarly implemented as a static class ENVI which contains wrappers around the existing `ENVIreader/writer` functions.

4. FUNCTIONALITY

This section will detail some of the main features of the class and provide examples of their usage. It is not intended as a full API reference, but as more of cursory look at common MATLAB operations on multidimensional data and their implementation in the Cube class.

4.1 Constructing Cubes

Given a hyperspectral data cube (or any 3-dimensional array), the construction of a Cube object is straightforward. For a simple example, the syntax

```
c = Cube(ones(10,10,16));
```

creates a Cube object with the given array and default metadata (vector of band indices 1 to 16 for `Wavelength`, vector of zeros for `FWHM`, and `Quantity` set to the string "Unknown"). Metadata can be specified to the constructor using the usual "Name, value" syntax used by many MATLAB functions. As an example, one could construct a mockup radiance Cube with actual wavelength metadata as

```
mockup = Cube(ones(10,10,25), ...
      'qty', 'Radiance', ...
      'wlu', 'nm', ...
      'wl', 501:525, ...
      'fwhm', 5*ones(1,25));
```

The possible arguments can be found from the documentation of the `Cube()` constructor method or by directly inspecting the argument parser class CubeArgs.

4.2 Slicing

Instead of the usual MATLAB slicing syntax for multidimensional matrices, the Cube class implements different methods for slicing different dimensions. While slightly more verbose, this makes code much more readable, especially when combining multiple operations on a single line.

- For spatial slicing, the current implementation has the method `crop(tl,br)`, which takes in the top left and bottom right corners of the desired area as 2-element vectors of the pixel coordinates, and returns the rectangular region defined by the corners. For more advanced region selection, users should employ the mask and unmask functionality shown in section 4.5. For a simple spatial crop of 20 pixels on each side of the image, one could supply the following syntax:

  ```
  cropped = cube.crop(...
    [20,20], ...
    [cube.Width-20, cube.Height-20])
  ```

  Note here the usage of the Cube properties `Width` and `Height` to extract the dimensions, which makes the code very readable at only a slight cost to terseness.

- The method `px([x,y])` selects individual pixels based on their pixel coordinates. It is possible to supply multiple coordinates, in which case the resulting Cube will be a list ($N \times bands$ matrix) of the spectra in those coordinates.

- For band selection, the method `bands()` takes in a vector of band indices (between 1 and the number of bands in the data) and returns a Cube object with the specified bands. The syntax allows for some more manipulations besides just selection of existing bands: For instance, it is possible to replicate bands by supplying the same index multiple times. There is currently no direct support for selecting wavelengths explicitly. Instead, `bands()` also accepts logical vectors for selecting bands, which allows selection of certain wavelengths by e.g. the following syntax using the `Wavelength` property of the Cube:

```
longer_wavelengths = ...
   cube.bands(cube.Wavelength > 1000);
```

Both syntaxes will result in Cubes that have their other properties appropriately modified, with only the wavelength and FWHM information corresponding to the selected band left preserved in the metadata.

For comparison, listing 1 demonstrates the amount bookkeeping required for selecting wavelengths from a dataset without the Cube class, and keeping the metadata synchronized at the same time. It also demonstrates the namespace pollution that is hard to avoid without more specialized data containers that reduce the need for separate variables for in-memory data management.

Listing 1. Selecting data by wavelength with plain MATLAB

```
wl   = 501:1000;              % mock wavelengths
fwhm = 5*ones(500);           % mock fwmh data
g    = rand(100, 100, 500); % random datacube

% Selecting bands corresponding to 600-800 nm range
idx = wl >= 600 & wl <= 800;
g2 = g(:, :, idx);
wl2 = wl(idx);
fwhm2 = fwhm(idx);
```

4.3 Arithmetic

Basic arithmetic (namely the operators `+,-,*,/`) is currently implemented by overloading the operators for the Cube class. The `Quantity` parameters of the argument cubes are by default naively combined to denote the new quantity, i.e. dividing a radiance cube by a radiance cube will produce a cube with quantity “radiance / radiance”. However, since MATLAB allows extra arguments to overloaded functions, it is possible to use the syntax `oper(a,b,quantity)` to supply the result quantity as an extra argument. As an example with the radiance Cubes, one might wish to calculate reflectance using the following syntax:

```
refl_cube = div(cube, white_ref_cube, ...
        'Reflectance');
```

The resulting Cube would then contain the data in `cube` divided elementwise by that of `white_ref_cube`, with the `Quantity` set to the string “Reflectance”.

In the current implementation, the operators are explicitly restricted to Cubes of the exact same dimensions (also erroring on any arrays that are not Cubes). This is in contrast to the normal MATLAB operators on numerical arrays, which in the 2017a version do automatic expansion of the argument arrays (using `bsxfun`). This is due to the fact that `bsxfun` does not necessarily replicate the arguments along a dimension that is sensible for a hyperspectral data cube. For instance, if one were to add a constant vector (like a bandwise correction to the spectrum) to a full data cube, the correctness of the result would depend on which of the three dimensions of the cube would match the length of the vector first, and could result in errors for cubes with two dimensions with the same length.

Other mathematical operations that one might want to apply on a single cube (such as multiplication by a scalar, logarithms etc.) can be applied by using the existing functions with the family of map functions detailed in section 4.4.

4.4 Function application

For more functionally inclined programmers, the Cube class implements methods for applying given functions on the Cube data without the need for explicit deconstruction of the Cube object. For applying functions expecting various dimensions of data, three different functions are implemented:

- The method `map(f, ...)` may be used to apply a given function `f` on the whole data cube. If the function changes the data in a meaningful way, the user is expected to supply the new metadata for the result as separate parameters using the Name, value syntax of the Cube constructor.
- `mapSpectra(f, ...)` applies the function `f` on the data after reshaping the hyperspectral data cube into a list of the spectra, and after application reshapes the result into the original spacial dimensions of the data (with possible change in the number of bands). This is equivalent to the functionality demonstrated in section 4.5, but using the full data cube instead of selected parts of it.
- `mapBands(f)` applies the function `f` to each band of the data cube by looping through each layer in turn, applying `f` and collecting the results. This provides an easy way to apply filtering operations that are not dependent on the wavelength, but due to the looping (which there is in this general form no easy way to avoid), it may be preferable to implement more costly filter directly on cubes and apply them using `map`.

4.5 Masking and unmasking

For selecting regions of interest (especially non-rectangular ones) from a Cube, the class implements a method called `mask`. Given a binary mask image (logical matrix) with a size matching the spatial dimensions of the Cube, it reduces the Cube to a $N \times bands$ matrix of pixel spectra selected by the mask, with N the number of True pixels in the mask image. The rationale for the list form of the output is compatibility with machine learning due to the fact that the $N \times bands$ matrix is of the form $samples \times features$ expected by most existing machine learning applications, which makes it very fast to use `mask()` to extract data for classification tasks.

The `unmask()` method is used for applying the reverse transformation: Given a mask image with N True pixels, it reorganizes an $N \times bands$ (or $samples \times features$) Cube into a full Cube with the spatial dimensions of the mask image, with zeros in the False mask regions and the data in the True region. If the data in the list is in the same order as returned by the corresponding `mask()` operation, applying `unmask()` after `mask()` with the same mask image results in having a mask applied on a data cube by zeroing any False entries on each layer.

This along with the `map()` methods (4.4) allows for very concise applications of existing algorithms on hyperspectral cubes. For instance consider the example workflow of applying PCA to masked part of the image in listing 2. For comparison, the same workflow without the abstractions provided by the Cube class is also shown.

Listing 2. Performing PCA on a subset of pixels

```
% g is a datacube with the h*w pixels and b bands
[h, w, b] = size(g);
```

```
% Construct a Cube from the data
c = Cube(g);
% Dummy mask that selects the diagonal
im = eye(c.Height, c.Width);

% Application of PCA using masks and map
c_scores = c.mask(im).map(scores).unmask(im);

% Helper function to extract the scores from PCA
function [y] = scores(x)
    [~, y] = pca(x);
end
```

4.6 File operations

For convenience, the implementation includes a separate class for wrapping `ENVIreader/writer` operations with those of the Cube class. The included ENVI class has methods for reading and writing ENVI files directly to Cube objects, filling in all the Cube metadata fields directly from the ENVI header file. A write wrapper method has also been implemented for directly writing both the data and metadata from a Cube object to an ENVI file. Other file formats can easily be added using similar wrappers for existing functionality.

4.7 Visualization

For visualization, the class implements a few plotting and image viewing methods to make data exploration easier. The main advantage over using the MATLAB visualizations directly is that the Cube class can utilize its metadata to assign most axis ticks, labels and titles appropriately in the produced figures. If found in path, they also employ the MATLAB version of Colorbrewer (Cobeldick, 2017) colormaps for better representation of the data. The main visualizations are provided by the following methods:

- The method `im(b)` displays a single band `b` as an image using the MATLAB function `imagesc`.
- `rgb(r,g,b)` displays the bands at indices `r, g` and `b` as a three-color image using MATLAB `imshow`.
- `plot()` plots the spectra using MATLAB `plot()`. It also automatically restricts the number of spectra to display in order to prevent MATLAB from freezing when trying to plot too many curves at once.
- `hist()` calculates histograms of each band layer in the Cube, combines them and displays them using `surf` in combination with a colormap for a versatile visualization.

The visualizations always return the Cube they were visualizing, which makes it easy to both extract the result of a Cube operation and visualize it using a single line of code. For example, the following line would visualize the first band of the cube, apply a crop and then display the same band in the cropped image in a new figure, while also extracting the result as a new Cube:

```
tl = [20,20]; % top left
br = [100, 100]; % bottom right
cropped = c.im(1).crop(tl, br).im(1);
```

This generally makes data exploration effortless and easy to integrate with follow-up scripting.

4.8 Provenance

Apart from the visualizations, all the Cube methods that operate on the data append a string representation of their action to the `History` property of the returned Cube. This allows those of us with less-than perfect memory of our command line usage to inspect the operations performed on the Cube objects in the MATLAB workspace by simply looking at the strings stored in the property of each object. In the current implementation, there is however no inbuilt way to store the history of each Cube in a file apart from saving the whole object to a file, which is in general a fragile procedure due to the way MATLAB handles object loading.

5. TESTING

The class constructor and methods have unit tests written for the using the MATLAB unit testing framework. Integration testing of method and constructor interoperations has not yet been implemented apart from some individual cases. The tests are included in the release version of the package, and can be run by users on their MATLAB installation to ensure compatibility with their given version of MATLAB. People interested in the framework are invited to submit test cases and bug reports to the author (preferably through pull requests to the Github repository, see section 6).

6. OBTAINING THE CODE

At the time of writing, version 0.8.0 of the `hsicube` framework is available from the authors Github page[1] under the MIT license. The package contains the methods needed to directly read ENVI files under the `ENVI` module, however this functionality requires one to have the `ENVIreader/writer` (Totir and Howat, 2010) in their MATLAB path, which is not included in the package and must be acquired separately. Similarly, for the nicer Colorbrewer colormaps (Cobeldick, 2017) the required package needs to be acquired separately and placed in path before utilizing the visualizations.

REFERENCES

Boggs, T., 2016. Spectral python. Software. Available from http://www.spectralpython.net.

Cobeldick, S., 2017. Colorbrewer: Attractive and distinctive colormaps. Software. Available from https://se.mathworks.com/matlabcentral/fileexchange/45208-colorbrewer–attractive-and-distinctive-colormaps.

Gerg, I., 2016. Matlab hyperspectral toolbox. Software. Available from https://github.com/isaacgerg/matlabHyperspectralToolbox.

Hoyer, S. and Hamman, J., 2017. xarray: N-d labeled arrays and datasets in python. *Journal of Open Research Software*.

Salmi, P., Eskelinen, M. A., Kremp, A. and Pölönen, I., 2017. Constructing absorbance spectra and abundance maps of micro- and nanoalgae by transmission hyperspectral imaging of liquid cultures on petri dishes. *PLOS One*. Submitted.

Totir, F. and Howat, I., 2010. Envi file reader/writer. Software. Available from https://se.mathworks.com/matlabcentral/fileexchange/27172-envi-file-reader-writer.

[1] `https://github.org/maaleske/hsicube`

PIII

PRACTICAL APPROACH FOR HYPERSPECTRAL IMAGE PROCESSING IN PYTHON

by

Leevi Annala, Matti Eskelinen, Jyri Hämäläinen, Aamos Riihinen, Ilkka Pölönen
2018

ISPRS TC III Mid-term Symposium ”Developments, Technologies and Applications in Remote Sensing” (pp. 45-52). International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume XLII-3. International Society for Photogrammetry and Remote Sensing. DOI: 10.5194/isprs-archives-XLII-3-45-2018

PRACTICAL APPROACH FOR HYPERSPECTRAL IMAGE PROCESSING IN PYTHON

Leevi Annala*, Matti A. Eskelinen, Jyri Hämäläinen, Aamos Riihinen, Ilkka Pölönen

Faculty of Information Technology, University of Jyvaskyla - leevi.a.annala@student.jyu.fi

Commission III, WG III/4

KEY WORDS: Python, Data analysis, Hyperspectral imaging, Image processing, Machine learning, Open source

ABSTRACT:

Python is a very popular programming language among data scientists around the world. Python can also be used in hyperspectral data analysis. There are some toolboxes designed for spectral imaging, such as Spectral Python and HyperSpy, but there is a need for analysis pipeline, which is easy to use and agile for different solutions. We propose a Python pipeline which is built on packages xarray, Holoviews and scikit-learn. We have developed some of own tools, MaskAccessor, VisualisorAccessor and a spectral index library. They also fulfill our goal of easy and agile data processing. In this paper we will present our processing pipeline and demonstrate it in practice.

1. INTRODUCTION AND MOTIVATION

Python is a go-to programming language of many scientists and it could also be good programming language for hyperspectral data analysis. It has advantage of being actively developed, free, open source programming language. In addition, since it looks like pseudocode, it is easy to learn and write. There are Python tools and packages for all kinds of users, and especially for scientists. There are specialized open source tools for hyperspectral data analysis like Spectral Python (Boggs, n.d.) and HyperSpy (de la Peña et al., 2017), but the scope of potential usage may be too narrow and the structure of such an specialized tool can be too strict for some purposes, for example for transferring data to machine learning algorithm and developing tools that work together with them.

In this paper, we utilize some general open source tools for different aspects of hyperspectral data analysis and determine if they are useful for analysing and visualising hyperspectral images. We also introduce some new tools and packages, which are our own work. We aim at providing the reader with a modular set of tools that can be used in many contexes. These tools are reusable elements, which work fine on their own and can be used for building more complex tools. The packages and tools will be evaluated using following questions: How easy is it to use? How agile is it? What can we do with it?

2. DIFFERENT ASPECTS OF HYPERSPECTRAL DATA ANALYSIS

In this section we will go through different aspects of hyperspectral data analysis and an example of how the selected tools can be used in these subjects. The example is divided into smaller examples and what has been done on previously is assumed to hold on to the new example. We go through the example in figures and in text, and the source code is included in the figures. The example problem is that we have a hyperspectral image of a forest and a dataset of two tree species, birch and pine, in that forest, and we want to use machine learning to differentiate one from the other. First we of course need to import all of the packages, like in figure 1.

```
import xarray as xr
import numpy as np
import pandas as pd
import holoviews as hv
from sklearn import svm
import sklearn
from sklearn.model_selection import GridSearchCV
import visacc
import maskacc

hv.notebook_extension('matplotlib')
```

Figure 1. Importing all necessary packages and declaring that Holoviews should use Matplotlib backend.

2.1 Handling hyperspectral data

For handling hyperspectral data, we recommend the xarray[1] package (Hoyer and Hamman, 2017). It provides multidimensional arrays and datasets with metadata. It is an actively developed open source project by the pydata team. The basic usage of xarray is relatively easy and for more advanced users it offers plenty of options for handling the data. Xarray’s basic idea is to have netCDF (Rew et al., 1997) compatible multidimensional array object in Python. NetCDF stands for network Common Data Form and the basic idea is that the netCDF file describes itself to the reader. Xarray is also easily extendable, which means that one can add new properties as they are needed.

Xarray supports reading spectral image formats like ENVI or TIFF, and other formats. For reading it uses Rasterio (Gillies et al., 2013–), which in turn uses GDAL (GDAL Development Team, 2018). Rasterio is a python toolbox developed solely to read and write geospatial data, and it does it well. GDAL (Geospatial Data Abstraction Library) is a lower level C++ library that translates geospatial raster and vector data.

When xarray has read dataset from file (see figure 2), it is either DataArray or Dataset. There are differences between the two, but

*Corresponding author

[1]Xarray can be installed with pip (`pip install xarray`) or conda (`conda install xarray`) Python package managers.

from now on we will assume that the data is in DaraArray format. DataArray has following properties (see figure 3):

- data, N-dimensional NumPy (Oliphant, 2006) or Dask (Dask Development Team, 2016) array,
- coords, dictionary of coordinate arrays, one array for each dimension of the data,
- dims, names of the dimensions,
- attrs, dictionary keeping track of other metadata,
- name, the name of the DataArray,

which follow the netCDF specification. These properties help in extracting data from the DataArray, since the user can use either index based lookups or label based lookups. For example, if we only had NumPy[2] array, we would only know the dimensions by index, but with DataArray we have names like latitude, longitude and wavelength[3]. Then we can extract data from DataArray like in figure 4 by telling it that we want to see data where latitude is between 39° N and 40° N, longitude is between 116° E and 117° E, and wavelength is between 400 nm and 700 nm.

```
cube = xr.open_dataarray(
        'C:/Users/lealanna/DATAA/vvkk2.nc'
        )
wavelength = [507.60, 509.50, 514.50, 520.80,
              529.00, 537.40, 545.80, 554.40,
              562.70, 574.20, 583.60, 590.40,
              598.80, 605.70, 617.50, 630.70,
              644.20, 657.20, 670.10, 677.80,
              691.10, 698.40, 705.30, 711.10,
              717.90, 731.30, 738.50, 751.50,
              763.70, 778.50, 794.00, 806.30,
              819.70, 833.70, 845.80, 859.10,
              872.80, 885.60]
cube.coords['wavelength'] = ('band', wavelength)
cube = cube.swap_dims({'band': 'wavelength'})
cube.values[cube.values<0]=np.nan
```

Figure 2. Here we read the cube, attach wavelength data to it and remove non-physical negative values.

```
print(cube)
```

```
<xarray.DataArray (wavelength: 38, y: 4120, x: 3930)>
array([[[ nan,  nan, ...,  nan,  nan],
        [ nan,  nan, ...,  nan,  nan],
        ...,
        [ nan,  nan, ...,  nan,  nan],
        [ nan,  nan, ...,  nan,  nan]],

        ...,
        [ nan,  nan, ...,  nan,  nan],
        [ nan,  nan, ...,  nan,  nan]],
        ...,
        [ nan,  nan, ...,  nan,  nan],
        [ nan,  nan, ...,  nan,  nan]]],
        dtype=float32)
Coordinates:
    * longitude   (longitude) float64 6.804e+06 ...
    * latitude    (latitude) float64 3.983e+05  ...
    band         (wavelength) int32 1 2 3 4 5 6 7 ...
    * wavelength  (wavelength) float64 507.6 509.5...
Attributes:
    res:          [ 1.  -1.]
    is_tiled:     1
    transform:    [1.00000000e-01  0.00000000e+00 ...
    ncols:        3930
    rows:         4120
    xllcorner:    398296
    yllcorner:    6804299
    cellsize:     0.1
```

Figure 3. Simple print-command to see what the cube holds inside.

[2]NumPy is in practice the Python standard array library.

```
cube.sel(latitude=slice(39,40),
         longitude=slice(116,117),
         wavelength=slice(400,700))
```

Figure 4. Here we use xarray's sel-method to extract the data we want.

There are also other useful functionalities of xarray DataArray. For example two or more arrays can be attached to each other with easy one line command, where the user only has to align the arrays by common dimension. Generally speaking, xarray handles dimensions well and altering and extracting data using them is generally quite easy. Xarray also handles missing data well and there is possibility to use Dask arrays to parallel compute.

Xarray fullfills our criteria of being easy to use and agile. It has a lot of functionality, enough to keep basic and advanced users satisfied most of the time.

2.2 Visualisation

For visualizing the xarray data, one excellent solution is Holoviews[4] (Stevens et al., n.d.). Holoviews is a visualization library that uses Bokeh (Bokeh Development Team, 2014), Matplotlib (Hunter, 2007) or Plotly (Plotly Technologies Inc., 2015) for showing images. All figures in this paper are produced with Holoviews using Bokeh or Matplotlib visualisation backends.

Basic idea of Holoviews is that visualizing of data should be easy and simple. If user wants to see anything, it should not take many lines of code. In our opinion, Holoviews succeeds in that goal. As we move on, one will see that all images in this paper are produced with less than four lines of code. One basic example of producing Holoviews image is to look at one band of a hyperspectral image like in figure 5.

Now that we have figured out how to visualise a single channel of an image, the next logical step is to want to visualise the entire multidimensional dataset. This is also easy. Holoviews supports multidimensional datasets very well and there are data backends that support multiple different data formats including xarray. As we can see in figure 6, more complex visualisation is easy to make. In the example we make a Holoviews dataset out of xarray DataArray, and tell Holoviews to make a series of images out of the dataset.

One of the properties of Holoviews is that one can make interactive figures using the Bokeh backend with no extra effort. By having Bokeh backend selected user can right away use interactive tools like zooming the image either by scrolling or drawing boxes on the image. A little more work is required for using hover, tapping or selection tools, which all can be programmed to do what the user wants them to do. An example of usage of tapping and selection tools are using them to select data for further analysis or activating other visualisation with them.

[3]Note, that the user can freely name the dimensions. The user is not stuck with these names.

[4]Holoviews can be installed with pip (`pip install holoviews`) or conda (`conda install holoviews`) Python package managers.

```
cube = xarray.DataArray(...)
hv.Image(cube.sel(wavelength=800,
                  method='nearest'))
```

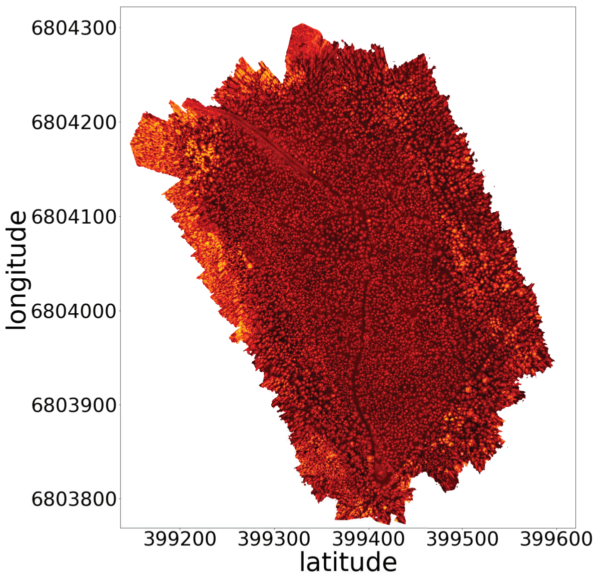


Figure 5. Here we produce a very simple Holoviews visualisation by telling Holoviews Image to use the xarray data we provide it. This is an image of a Finnish forest.

A good platform for using Holoviews is Jupyter Notebook (Kluyver et al., 2016). Jupyter Notebook is a web application where user can code in Python and output images and write narratives between code blocks. The user has to activate Holoviews by importing it and declaring the visualisation backend like in figure 1. When one is visualising figures in Jupyter Notebook, it is possible to fine tune figures, by using *output cell magic* and Holoviews *opts*. We use output cell magic and opts in figure 6, where lines starting with % or %% are the cell magic lines. In the example of the figure we tune the size of the font and the size of the image. This fine tuning is absolutely necessary if one needs to produce figures for a publication or needs good looking images for any reason. Matplotlib backend is better suited for publication quality figures.

Holoviews is purely a visualisation library. The user can make data move between two images in the same visualisation, but the developers have not build a way to get this data for further use and the only way of getting a data output is by coding it. However, once coded, these background processes are relatively easy to attach to an image. Holoviews is an open source project and it is developed by the ioam team. Holoviews is easy to use and it can be bended to do many things. It makes beautiful images, and in all is an excellent choise for visualisation.

2.3 Masking and visualizing xarray

Using xarray and Holoviews together is made easy by Holoviews developers. Xarray is one of the available backends for Holoviews. That means, one can easily produce an image from xarray using Holoviews. There is still some difficulties involved, and to address those difficulties, we use xarrays extendability. Making an extension to xarray in figure 7 is done by making a Python class and declaring it as dataset or DataArray accessor.

```
%%opts Image [fontsize={{'title':15, \
                        'xlabel':15, \
                        'ylabel':15, \
                        'ticks':15}},\
        fig_size=350]

ds = hv.Dataset(cube,
                vdims=['Value'])
ds.to(hv.Image,
      kdims=['x', 'y'],
      dynamic=True)
```

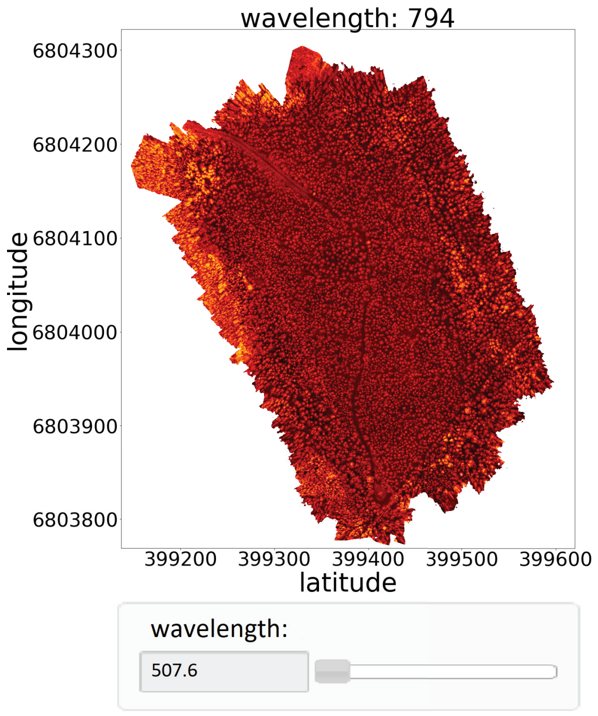


Figure 6. Here we make a more complicated Holoviews visualisation by using Holoviews dataset. From using Dataset, we get a slider that goes through the wavelength bands.

These extensions are relatively easy to make and can extend xarray's functionalities to anything one might want it to do, within reasonable limits.

```
@xr.register_DataArray_accessor('cat')
class CatAccessor(object):
    def __init__(self, xarray_obj):
        self._obj = xarray_obj
        self.cat = 'a_cat'
```

Figure 7. Extending xarray with a simple Accessor. Here we declare that CatAccessor is a DataArray accessor and define it.

We have developed two DataArray accessors, MaskAccessor and VisualisorAccessor[5]. The reason for developing both of these tools is that we want to use more complicted background interactivity tracking with Holoviews and get the data out of the visualisation.

MaskAccessor is a general masking tool for xarray, and the main function of it is to help collect data points to further analysis, such as machine learning or modelling. It provides an interface

[5]These tools are available at our groups github page `http://github.com/silmae`

for selecting pixels in n-dimensional datasets. In figure 8 we see that the accessor is initiated when one imports the xarray and the accessor. After that every DataArray has the property, and the user can use the accessor by calling it by name.

```
import xarray as xr
import maskacc

cube = xr.DataArray(...)
cube.M.dims
```

Figure 8. When the accessor is imported, every xarray DataArray created after that has the accessor attribute.

The mask dimensions are set at the initialisation to be the first two dimensions of the DataArray, but there is the reset method that is used to change the dimensions, as we see on figure 9. One can also initialise the mask here or just assign a new mask afterwards. The MaskAccessor class checks that the shape of the mask is correct.

```
import numpy as np
cube.M.reset(dims=['a', 'b'],
             matrix=[[0,1,0,1],
                     [1,0,1,0],
                     [0,1,0,1]])

# OR
cube.M.reset(dims=['a', 'b'])
cube.M.mask = np.array([[0,1,0,1],
                        [1,0,1,0],
                        [0,1,0,1]])
```

Figure 9. Different ways of assigning a specific matrix as the mask.

On figure 10 one can see four different selection methods to set mask on individual points.

```
# Select
cube.M.select([0,0])
cube.M.select([(0,2),(1,1)])

# Unselect
cube.M.unselect([(0,2),(1,1)])

# All to ones
cube.M.selected_ones()

# All to zeros
cube.M.selected_zeros()
```

Figure 10. Different selection methods for MaskAccessor.

Finally, on figure 11 there is three different methods to get the mask or masked data.

```
# Get the mask as xarray.DataArray
cube.M.mask_as_xarray()

# Get the masked points as xarray.DataArray
cube.M.where_masked()

# Get the masked points as a list
cube.M.to_list()
```

Figure 11. Methods for getting data out of MaskAccessor and underlyind DataArray.

VisualisorAccessor is a hyperspectral imaging specific visualising tool for xarray and MaskAccessor. It is designed to make basic visualizations of xarray DataArray and MaskAccessor mask with easy one-line commands. For example the image in figure 6 can now be produced with the one line code of figure 12. It is also easy to add visualisations like this to the VisualisorAccessor.

```
cube.visualize.basic(sliders = ['wavelength'])
```

Figure 12. The visualisation on figure 6 can be done with one line code with VisualisorAccessor.

We have implemented three chooser functions, which access the mask and select or unselect pixels. They are called Point Chooser, Box Chooser and Spectre Chooser. Spectre Chooser and Box Chooser use Bokeh’s box drawing tools for selecting which pixels are chosen and Point Chooser uses tap tool. Example uses of the Choosers is on figure 13, and screenshots of the Choosers are on figures 14 (Point Chooser), 15 (Box Chooser) and 16 (Spectre Chooser).

```
layout_box = cube.visualize.box_chooser()
layout_point = cube.visualize.point_chooser()
layout_spectre = cube.visualize.spectre_chooser()
```

Figure 13. VisualisorAccessor has three different chooser tools.

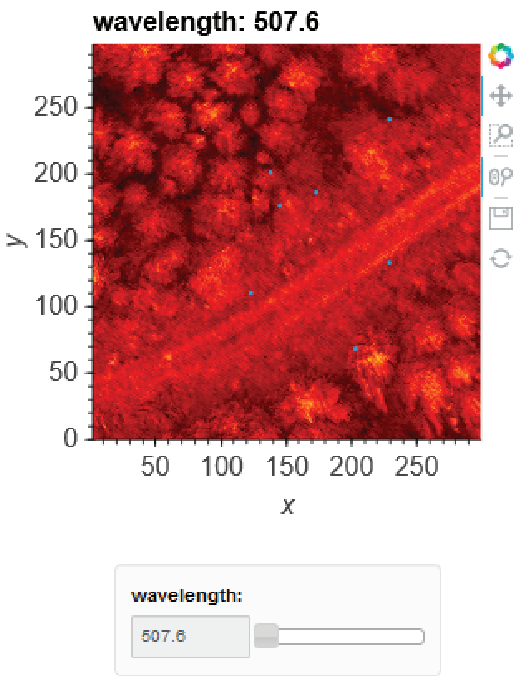


Figure 14. Screenshot of the point chooser.

Finally there is a histogram method (figure 17), that calculates histograms for each bands and shows those histograms side by side. This is translated from hsicube (Eskelinen, 2017) MATLAB package to Python.

2.4 Machine learning

Machine learning can be handled using scikit-learn[6] package (Pedregosa et al., 2011). The main idea of scikit-learn is to make

[6]Scikit-learn can be installed with pip (`pip install sklearn`) or conda (`conda install sklearn`) Python package managers.

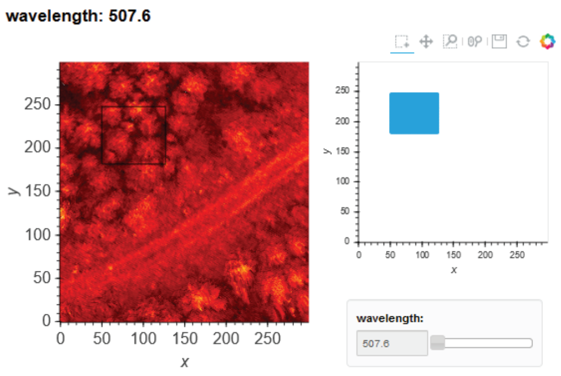


Figure 15. Screenshot of the box chooser.

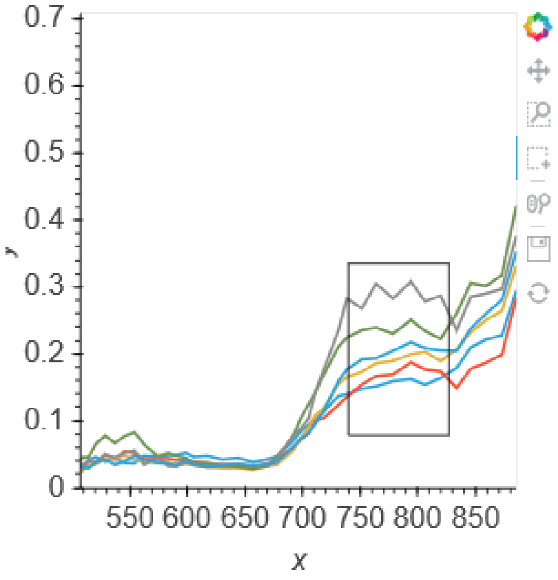


Figure 16. Screenshot of the spectre chooser.

simple and efficient tools for data analysis. Part of the simplicity is documentation, and with scikit-learn it is done well. There is flowchart for finding a suitable estimator, and every estimator is documented so well, that one can easily learn to use them decently.

Another thing we want to point out is the variety of implemented algorithms. Every well established machine learning algorithm can be found. Still, there are no duplicates, and the user does not have to worry about competing implementations, and the API is consistent through the algorithms.

Other useful properties are flows, parallel computing, fine tuning. The user can relatively easily make a workflow, that preprosesses data, does cross-validation on desired estimators with desired parameters and returns the estimator, that seems to produce the best result. The basic forms of the estimators are simple, but there are multiple parameters that one can use to fine tune the estimator.

The usage is simple since the algorithms are well documented and their API is simple, yet agile. Scikit-learn is also free and open source, and it is developed by scikit-learn team. It fullfills

```
result = cube.visualize.histogram(band_dim = 'band')
hist_image = result[0]
hist_counts = result[1]
bin_edges = result[2]
hist_image
```

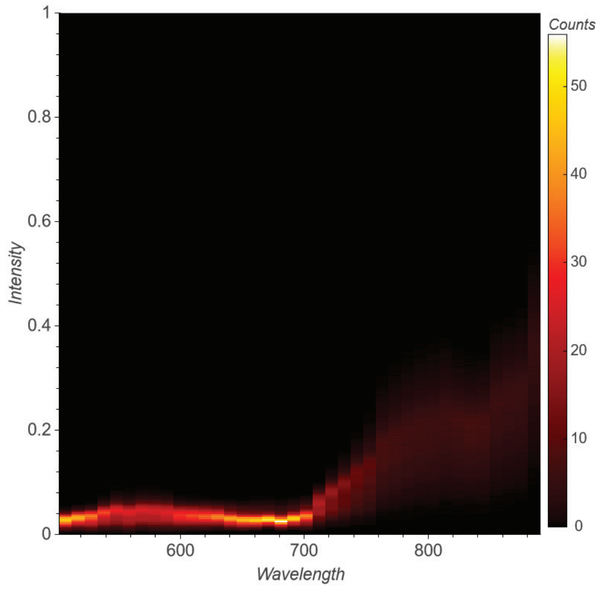


Figure 17. Visualisation of the histogram. The histogram tool returns an image of the histogram, the values of the histogram and the bin edges.

our criteria of being easy to use. It is agile in a way that user can make own flows through the algorithms and the user can fine-tune the algorithms as much as is needed.

Now we can use machine learning on our example problem. First, in figure 18 we use pandas[7] (McKinney, 2010) for reading the tree dataset and plot it over one of the bands in our cube.

Now we can use the tree coordinates to train a machine learning algorithm to recognise birch from pine. We take 30 * 30 box around every tree and calculate histogram of the box. These histograms are used to train the algorithm. We also have to make nan-values zero for this. In figure 19 we use VisualisorAccessor to make the histograms and goal vector and prepare them for machine learning.

In figure 20 we train the machine learning algorithm. For this example we are using support vector machine algorithm. We also do cross-validiation with GridSearchCV. Both of these functions are functions from scikit-learn. Then we print out the results, and that tells us the best accuracy score[8] and the best parameters.

In figure 21 use the predictor to predict the species of a 30x30 histogram, that is made from a hyperspectral image of a tree. From the result we could then interpret wheter the estimator estimates the histogram as a pine or a birch.

[7]Pandas is in practice the Python standard for tabular and Excel type data.

[8]Note, that the score should not be taken too seriously, since this is a toy example, and birch and pine are really easy to recognise from eatch other.

```
trees_panda = pd.read_csv('trees_panda_kuva_2',
                          index_col=0)
trees_panda = trees_panda.sort_values("I"). \
drop_duplicates(["I", "P"],
keep="first")
points = hv.Points(trees_panda,
kdims=['I','P'])
hv.Image(cube[20]) * points
```

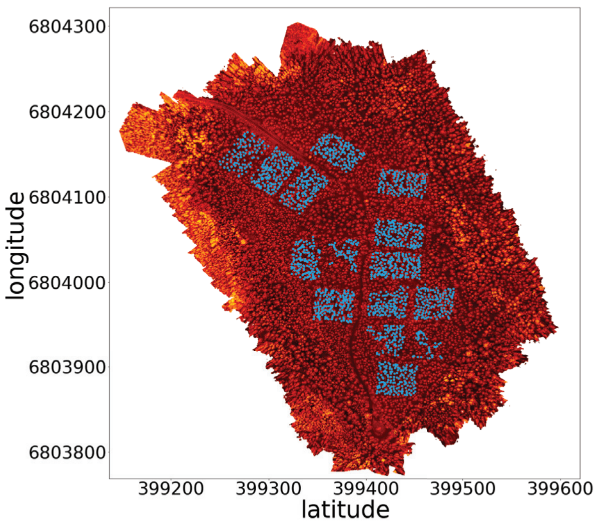


Figure 18. Visualisation of the trees over the image. In this block we read tree data as pandas DataFrame and visualise the trees on the top of one band of the cube.

```
cube.values = np.nan_to_num(cube.values)
X_list = []
size = 30
cellsize = float(cube.attrs['cellsize'])
add = cellsize * size / 2
bin_edges = np.arange(0, 1, 1/20)
i = 0
for puu in trees_panda.values:
print(i)
i = i+1
x_coord = puu[1]
#print(x_coord)
y_coord = puu[2]
#print(y_coord)
cropped = cube.sel(y=slice(y_coord + add,
                          y_coord - add),
                  x=slice(x_coord-add,
                          x_coord+add))
_, hist, _ = cropped.visualize.\
             histogram(
           bin_edges=bin_edges,
                     flag='linear',
                     show_plot=False,
                     band_dim="wavelength"
           )
hist_flat = np.array(hist).flatten()
X_list.append(hist_flat)
X = np.array(X_list)
y = np.array(trees_panda['pl'])
```

Figure 19. Calculating histograms and preparing data for machine learning algorithm.

2.5 Other aspects

Other notable libraries for hyperspectral data analysis are already mentioned Bokeh for advanced visualisation, scikit-image (van der Walt et al., 2014) for image data analysis and TensorFlow (Abadi et al., 2015) and Keras[9] (Chollet et al., 2015) for deep learning.

```
svc = svm.SVC()
parameters = {'kernel':['linear',
                        'poly',
                        'rbf',
                        'sigmoid'],
          'C':[10**i for i in range(-5,4)]}

clf = GridSearchCV(svc, parameters, n_jobs=20)
clf.fit(X,y)
clf_best = clf.best_estimator_
print(clf.best_score_)
print(clf.best_estimator_)
```

```
0.965723612622
SVC(C=0.1, cache_size=200, class_weight=None,
coef0=0.0, decision_function_shape='ovr',
degree=3, gamma='auto', kernel='poly',
max_iter=-1, probability=False,
random_state=None, shrinking=True,
tol=0.001, verbose=False)
```

Figure 20. Here we train the machine learning algorithm and print out the result. caption=Results of the training. Here we see that best estimator predicts correctly 96.6% of the time.

```
tree_1_pred = clf_best.predict(X_new.reshape(1, -1))
```

Figure 21. Here we use the estimator.

Bokeh is a package that has been on the rise in 2017. Bokeh makes interactive Python visualisations, using JavaScript. It is a backend of Holoviews, and if one wants to understand Holoviews deeply, this is one place to look at. Bokeh visualisations are generally quite beautiful, but it comes with expence of computational complexity and increased memory usage.

Scikit-image is a sister package of scikit-learn. Scikit-image is focused on computer vision and image processing. The same advantages as with scikit-learn apply here. The API is consistent and simple and the wide variety of algorithms is well curated.

TensorFlow and Keras are deep learning libraries. TensorFlow is considered to be the state of the art at this field, but the syntax is difficult and learning curve extremely steep. Keras uses TensorFlow as a backend, and offers simpler syntax. If one is a beginner on deep learning, Keras is a library to more easily get started, but as one is becoming more advanced user, TensorFlow's flexibility and increased tuning possibilities start becoming more attractive.

3. CONCLUSIONS AND FURTHER WORK

We have gathered and further developed an agile and easy to use pipeline for hyperspectral data analysis in Python. The tools we have investigated have wide range of advantages such as simple API:s, variety of different implementations, back ends and tools and extendibility.

In addition to that, Python programming language has large user base and active developer community, which guarantees that Python keeps up with needs of scientists. The packages mentioned in this paper are all actively developed and thoroughly tested.

[9]These tools can also be installed with pip or conda.

Also, especially xarray and Holoviews are good Python tools for hyperspectral data processing and visualisation. These tools seem like they are made for this use, but they still provide the generality of non-specialised tools. Compared to HyperSpy end Spectral Python our solution is much more modular and open to extending with new blocks.

Finally we would like to suggest, that in the context of using Python in hyperspectral data analysis, there is need for developing a graphical user interface that uses these tools and finding out best practises for utilising deep learning algorithms. We are starting to develop the graphical user interface in this summer.

Deep learning algorithms are becoming more and more attractive when there is more and more computational power available. The algorithms are computationally intense, but when they are used correctly they provide strikingly good results. These algorithms can be applied on many of the problems on the field of hyperspectral data analysis, such as object recognition, classification or for example analysing the health of a crop.

On this specific toolset there is work to do with parallelisation, since the datasets are huge and paralellisation would make the computations faster.

We have also started to develop a Python library for spectral indices, and are quite far in it already. The leading principle of our implementation is to make a simple implementation of every index on website indexdatabase.de, and wrap the implementation lightly with features that help in the usage. The point was to make the indices easily computable, so that the user could easily use a loop to go through the indices.

One thing we need to define was the API for selecting bands. For many of the indices, they are not defined for exact wavelengths like 745nm, but rather for red light and user needs to define this as he/she wishes. This is for now done by declaring the defaults in form of a Python dictionary. The other thing to consider is how is a band selected. Is it selected only if there is a clear match, the indice wants wavelength 500nm and our data has exactly that or is there room for approximation? If the used data is in format of xarray DataArray or Dataset, then it is possible to use the xarrays nearest neighbor -selection like in figure 5, otherwise one needs to implement their own selector. Once the index library is initialised like in figure 22, one can loop through the indices and find all the indices that can be computed on the dataset like in figure 23. Then the user can plot all possible indices with Holoviews like in figure 24.

```
from pyspindl import Indices, selectors
defaults = {'NIR': 815.7,
            'GREEN': 544.2,
            'RED': 595.3}
defaults.update(
        {k: defaults['RED'] for k in ['Red', 'R']}
        )
defaults['G'] = defaults['GREEN']
#Without defaults, we can not calculate some indices.
indices = Indices(selectors.from_xarray(
                  'wavelength',
                  method='nearest',
                  tolerance=8.0
                  ),
                defaults=defaults)
```

Figure 22. The initialisation process of spectral indices library. This is still work in progress.

Other thing we we are considering in developing this package is bands. How are they defined? There is big difference between a camera that has the same response on a interval around the middle value and camera that has more gaussian response. These differences should somehow be accounted for with software. The response function could be used in selection, and inbetween values coud be interpolated from two or more bands based on their responses. The response function is definately important in presicion applications and this problem needs to be solved.

```
matches = dict()
for iname, ifunc in indices.items():
    try:
        matches[iname] = ifunc(cube_cropped)
        # The following is necessary to
        # remove indices that result
        # only in +inf, -inf and NaN
        if not np.any(np.isfinite(matches[iname])):
            matches.pop(iname)
            continue
        # We have now built a dictionary
        # of index names and corresponding data.
        matches[iname].coords['index_name'] = iname
        # We also want to clean up
        # unnecessary coordinates, if any remaim
        for coordinate in ['band',
                           'fwhm',
                           'wavelength']:
            if coordinate in matches[iname].coords:
                matches[iname] = matches[iname]. \
                                 drop(coordinate)
    except (KeyError, TypeError, NameError):
        pass
print(str(len(matches))+'␣matching␣indices␣found.')
```

Figure 23. We loop through the indices, and take those that are sensible.

```
%%output size = 250
%%opts Image [invert_yaxis=True] (cmap='Spectral')
dataset = hv.Dataset(prettyfield,
                     kdims=['index_name', 'x', 'y'],
                     vdims='Index')
dataset.to(hv.Image,
           kdims=['x', 'y'],
           dynamic=True).hist()
```

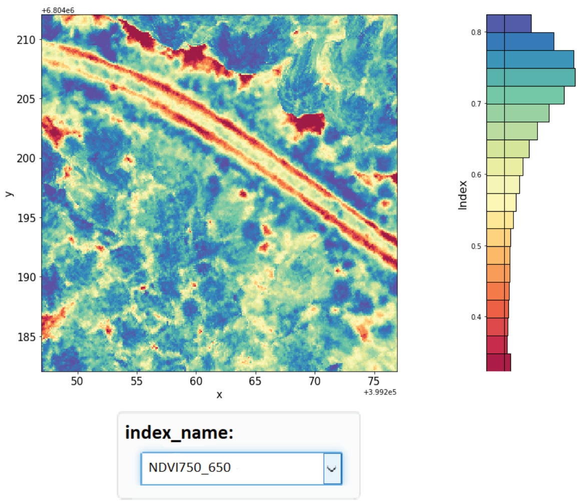


Figure 24. All indices in a dropdown menu. Dropdown menu comes from the use of Holoviews Dataset.

REFERENCES

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X., 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Official website: `https://www.tensorflow.org/`.

Boggs, T., n.d. Spectral python. Source code available at `https://github.com/spectralpython`.

Bokeh Development Team, 2014. Bokeh: Python library for interactive visualization. Official website: `http://www.bokeh.pydata.org`.

Chollet, F. et al., 2015. Keras. Source code available at `https://github.com/keras-team/keras`.

Dask Development Team, 2016. Dask: Library for dynamic task scheduling. Official website: `http://dask.pydata.org`.

de la Peña, F. et al., 2017. hyperspy/hyperspy: Hyperspy 1.3.

Eskelinen, M. A., 2017. Software framework for hyperspectral data exploration and processing in matlab. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-3/W3, pp. 47–50. Source code available at `https://github.com/silmae/hsicube`.

GDAL Development Team, 2018. Gdal - geospatial data abstraction library, version 2.2.3. Official website: `http://www.gdal.org`.

Gillies, S. et al., 2013–. Rasterio: geospatial raster i/o for Python programmers. Source code available at `https://github.com/mapbox/rasterio`.

Hoyer, S. and Hamman, J., 2017. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*. Source code available at `https://github.com/pydata/xarray`.

Hunter, J. D., 2007. Matplotlib: A 2d graphics environment. *Computing In Science & Engineering* 9(3), pp. 90–95. Source code is available at `https://github.com/matplotlib/matplotlib`.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J., Grout, J., Corlay, S., Ivanov, P., Avila, D., Abdalla, S. and Willing, C., 2016. Jupyter notebooks – a publishing format for reproducible computational workflows. pp. 87 – 90.

McKinney, W., 2010. Data structures for statistical computing in python. In: S. van der Walt and J. Millman (eds), *Proceedings of the 9th Python in Science Conference*, pp. 51 – 56.

Oliphant, T., 2006. A guide to numpy. Source code available at `https://github.com/numpy/numpy`.

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

Plotly Technologies Inc., 2015. Collaborative data science. Official website: `https://plot.ly`.

Rew, R. K., Davis, G. P., Emmerson, S. and Davies, H., 1997. NetCDF User’s Guide for C, An Interface for Data Access, Version 3.

Stevens, J.-L., Rudiger, P. and Bednar, J. A., n.d. Holoviews. Source code available at `https://github.com/ioam/holoviews`.

van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., Gouillart, E., Yu, T. and the scikit-image contributors, 2014. scikit-image: image processing in Python. *PeerJ* 2, pp. e453. Source code available at `https://github.com/scikit-image/scikit-image`.

PIV

MINIATURE MOEMS HYPERSPECTRAL IMAGER WITH VERSATILE ANALYSIS TOOLS

by

Roberts Trops, Anna-Maria Hakola, Severi Jääskeläinen, Antti Näsilä, Leevi Annala, Matti Eskelinen, Heikki Saari, Ilkka Pölönen, Anna Rissanen 2019

Proceedings of SPIE Volume 10931 : MOEMS and Miniaturized Systems XVIII; 109310W (pp. 109310W). SPIE conference proceedings, 10931. SPIE, The International Society for Optical Engineering. DOI: 10.1117/12.2506366.

Miniature MOEMS hyperspectral imager with versatile analysis tools

Roberts Trops[a], Anna-Maria Hakola[b], Severi Jääskeläinen[b], Antti Näsilä[a], Leevi Annala[b], Matti A. Eskelinen[b], Heikki Saari[a], Ilkka Pölönen[b], and Anna Rissanen*[a]

[a]VTT Technical Research Centre of Finland, Espoo, Finland
[b]Faculty of Information Technology, Jyväskylä University, Jyväskylä, Finland

ABSTRACT

The Fabry-Perot interferometers (FPI) are essential components of many hyperspectral imagers (HSI). While the Piezo-FPI (PFPI) are still very relevant in low volume, high performance applications, the tunable MOEMS FPI (MFPI) technology enables volume-scalable manufacturing, thus having potential to be a major game changer with the advantages of low costs and miniaturization. However, before a FPI can be utilized, it must be integrated with matching optical assembly, driving electronics and imaging sensor. Most importantly, the whole HSI system must be calibrated to account for wide variety of unwanted physical and environmental effects, that significantly influence quality of hyperspectral data. Another challenge of hyperspectral imaging is the applicability of produced raw data. Typically it is relatively low and an application specific software is necessary to turn data into meaningful information. A versatile analysis tools can help to breach the gap between raw hyperspectral data and the user application. This paper presents a novel HSI hardware platform that is compatible with both MFPI and PFPI technologies. With an MFPI installed, the new imager can have operating range of $\lambda = 600 - 1000$ nm with FWHM of $15 - 25$ nm and tuning speed of < 2 ms. Similar to previous imager in Ref. 1, the new integrated HSI system is well suited for mobile and cloud based applications due to its small dimensions and connectivity options. In addition to new hardware platform, a new hyperspectral imaging analysis software was developed. The new software used in conjunction with the HSI provides a platform for spectral data acquisition and a versatile analysis tool for a processing raw data into more meaningful information.

Keywords: Fabry-Perot interferometer, hyperspectral imager, MOEMS, VNIR, data analysis

1. INTRODUCTION

The advances in Fabry-Perot (FPI) interferometer technology has led the development of new miniaturized micro-spectrometers. These micro-spectrometers can be integrated in portable mobile devices, such as smartphones and tablets as well as specialized industrial equipment. The possibility to non-intrusively analyze spectral information of objects with mass producible mobile hyperspectral device is something that has applications in many different areas, such as agriculture, medicine, remote sensing and waste recycling.

The typical way of developing FPI based spectrometers involves selection FPI that offers best performance for the particular use case. The mechanics, optics and electronics are designed to adapt to selected FPI and imaging sensor. Unlike the previous imagers the new hardware described in this paper can be configured by interchanging FPIs and optics. With this feature, the capabilities of HSI hardware are extended and the same hardware platform can be used in more applications. Another challenge of hyperspectral imaging is the analysis of hyperspectral data cubes. The hyperspectral data cube contains the information about reflectance of light in certain wavelength range, yet unique spectral features might not be immediately apparent to the user. Each pixel in image has its own spectral response, but it is difficult to visualize and perceive such large amounts of data at once. In addition the spectra of each pixel might not even be relevant in most cases. Instead the user is typically interested in abundance or just existence of certain unique spectral features and the spatial information. This makes use of microspectrometers more challenging. It is not clear if the raw hyperspectral data contain

Further author information: (Send correspondence to Roberts Trops*)
Roberts Trops: E-mail: roberts.trops@vtt.fi, Telephone: +358-4017-6004-9, www.vtt.fi

any meaningful information. Furthermore, if there are any unique spectral features it is important to visualize in which ares of image these features occur.

Overall, it is clear that the emerging technologies for microspectrometers require a versatile and low cost hardware development platform as well as a data analysis tool, that would help the user to easily extract the most significant information out of hyperspectral data.

2. FABRY-PEROT INTERFEROMETER

Fabry-Perot interferometer is a type of optical measurement device, that utilizes two parallel mirrors to produce certain spectral response. There two main types of FPIs that VTT is developing. These types are MOEMS FPI (MFPI) and Piezo-actuated FPI (PFPI) both of which posses different inherent advantages, when used in hyperspectral imaging applications. Both types are electrically tunable, yet each of them employ a different actuation method. Typically the selection of FPI type for HSI is based on their general features. These features are summarized in table 1 and described in Ref. 2. Overall, the PFPIs with larger optical apertures are better for high performance applications where high signal-to-noise ratio (SNR) is essential. In contrast, MFPIs are excellent for volume scaling and applications where low sensor cost is necessary.

Table 1: General features of MFPI and PFPI technologies.

Technology	PFPI	MFPI
Manufacturing	Assembled structure for small-to-medium volumes	Mass-producible wafer-level processes scalable to volume manufacturing
Optical apertures	Up to 24 (mm) allow high throughput of light	Optical apertures of 2–4 (mm)
Passband tuning	Wide tuning range and easy mirror customization	Tuning range determined by mirrors

The MEMS FPI has a near-zero loss quartz substrate for visible range (wavelengths approximately $400 - 800$ nm). The figure 1 shows the structure of an MFPI chip. A drive voltage is applied to electrodes of MFPI to change the passband. If the drive voltage is exceeded, the mirrors in MFPI are pulled together by attractive electrostatic force created by the electric field between mirrors. This is called pull-in and it is typically non-reversible or not easily reversible state, depending on the conditions the pull-in occurred. Even chips that are manufactured on the same wafer will likely have different maximum drive voltage values. For practical purposes pull-in means that for the MFPI can no longer be used. This is typically avoided by charecterizing MFPI chips and settings hardware or software limits on voltage applied to the MFPI.[3]

Unlike the PFPIs the MFPIs can be mass-produced by using modern semiconductor fabrication techniques. This allows the chips to be made at much lower price. The MFPI chips are first manufactured on wafers, then diced and finally packaged. Figure 2 shows diced MFPI chips in a tray and a single packaged PFPI. The packaging adds mechanical protection, electrical interface and additional circuitry for FPIs.

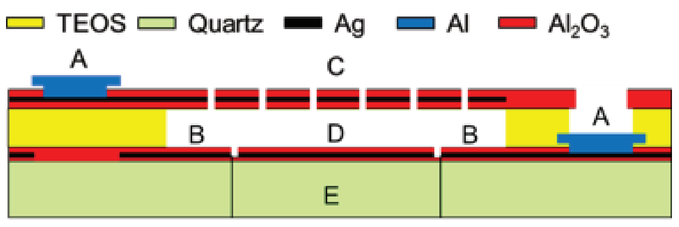


Figure 1: Structure of the MFPI. A: Contact pads; B: Actuation electrode; C: Upper movable mirror; D: Lower fixed mirror; E: Optical aperture.[1]

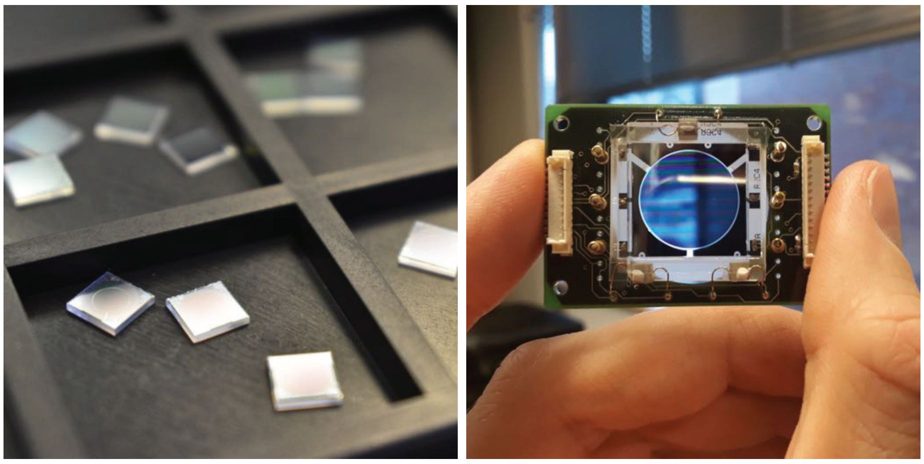


Figure 2: Unpackaged 5x5 mm MFPI chips (left) and packaged PFPI (right).

3. HYPERSPECTRAL IMAGER CONCEPT

This section describes a hyperspectral imaging hardware platform, that was developed based on MFPI and PFPI technologies. This platform is developed to be pseudo-autonomous system, that has integrated embedded Linux computer, that handles data acquisition, pre-processing, storage and transferring of data. In addition, the Linux computer has Ethernet, USB and WiFi connectivity, which can be used to add additional instruments or transfer image data. In addition to interchangeable FPIs, the hardware allows different optical assemblies.

3.1 Electronics

The new hyperspectral imager electronics design is partly based on previous hand-held imager design described in Ref. 1. The main components are camera module, which has MT9P031 5MP 12-bit CMOS sensor, Linux embedded computer and FPI controller. The camera module is connected to Linux computer via USB. The images are acquired using Python API. Similar to previous camera, the new design uses a controller board to interface between processing unit and respective FPI installed in imager.

3.2 Mechanics and optics

The optomechanical part of imager seen in figure 3 consists of aluminum baseplate, that holds optics, MFPI and sensor. In addition, it provides a way to focus and adapt the optics for different applications. The 3D printed case of imager can be seen in 4. It was designed to allow easy access to optomechanics, enclose electronics and provide tripod mounting interface. The table 2 summarizes two different PFPI and MFPI configurations, which the new HSI concept supports. These parameters are the most commonly used to describe FPIs, as well as select them for certain application, therefore having a hardware platform that is compatible with both types is a major advantage.

3.3 System calibration

After assembling HSI, it is necessary to perform system calibration. The monochromator setup shown in figure 5 is used to measure and calculate the spectral response of each pixel in camera sensor by gradually changing the wavelength of monochromatic light and the passband of FPI. Light from halogen lamp is passed into monochromator. Then, monochromatic light is passed into a integrating sphere, which has a reference detector and HSI attached. The passband of FPI and monochromator output are changed by software script running on PC. In addition, PC captures the images and calculates values for calibration table.

Table 2: Hardware configurations of HSI.

FPI type	*PFPI*	*MFPI*
Spectral range (nm)	450 – 850	600 – 1000
Spectral resolution (nm)	7 – 25 @ FWHM	15 – 25 @ FWHM
Settling time* (ms)	10	2
Clear aperture (mm)	14	2.5

*Settling time for small wavelength steps (typ. <10 nm).

3.4 Measurement calibration

In order to capture a valid hyperspectral data cube, first it is necessary to perform measurement calibration sequence. The calibration sequence consists of capturing the dark reference and a white reference. Measurement calibration has to be repeated in case of changes in the light source or significant changes in the temperature of imaging sensor and FPI. White reference target is typically a diffused target, that has uniformly high reflectance over whole measured wavelength range. The dark reference target is typically a highly absorbing material, that is used to fully cover the aperture of imager, while dark signal level is recorded.

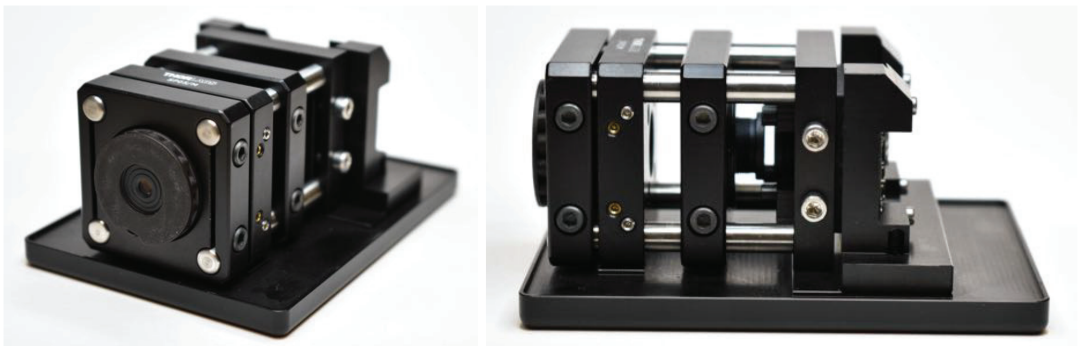


Figure 3: Interchangeable optomechanics mounted to baseplate.

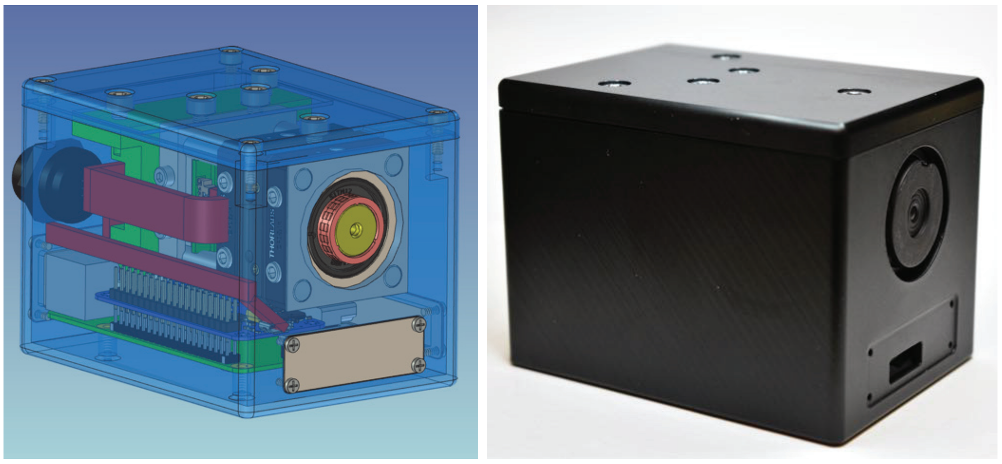


Figure 4: 3D CAD model and assembled version of hyperspectral imager.

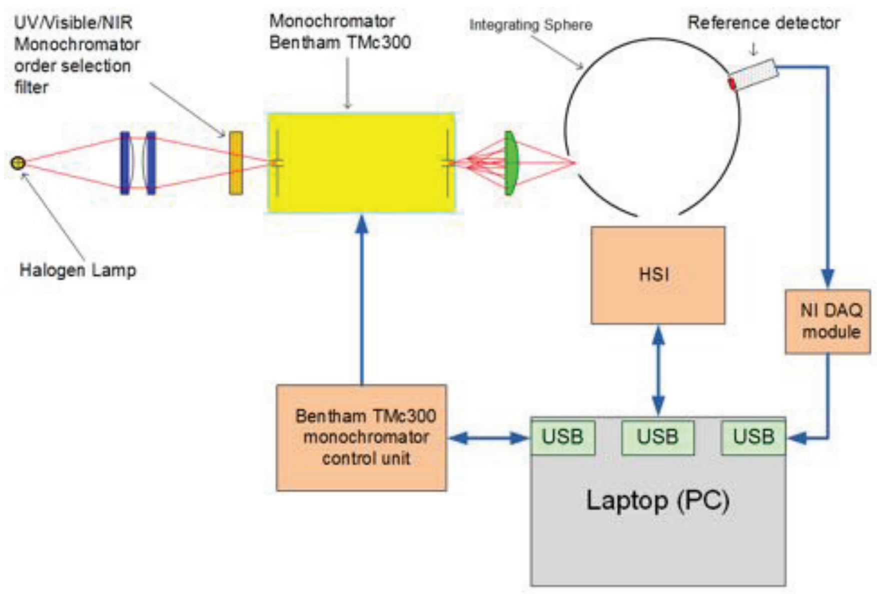

UV/Visible/NIR Monochromator order selection filter
Monochromator Bentham TMc300
Integrating Sphere
Reference detector
Halogen Lamp
HSI
NI DAQ module
Bentham TMc300 monochromator control unit
USB
USB
USB
Laptop (PC)

Figure 5: Monochromator calibration setup for HSI.

4. CUBEVIEW SOFTWARE

CubeView is a software, created by the Spectral Imaging Laboratory from the University of Jyväskylä, Finland. It's main purpose is to offer an easy to use and simple, but scientifically accurate interface for spectral imaging and spectral analysis. Software will be released under MIT license and one of the main strengths compared to other similar softwares is the compatibility with GeniCam standard. Other worth to mention features are the different ways to use CubeView: with devices it provides easy and fast imaging and without devices it can be used for analyzing an existing spectral data cubes.

4.1 Functionality

The essence of CubeView is the deep understanding of user requests and needs by using design thinking methods through the development phases. User requests and needs were based on Spectral Imaging Laboratory's past and ongoing research. The core of CubeView uses Spectral Imaging Laboratory's python libraries, `fpipy`,[4] `pyspindl`[5] and `spectracular`. CubeView has three main functionalities: LiveView, CubeView (imager) and CubeView Analyze.

LiveView is part of LiveView imager and it's main function is to offer video stream from the hyperspectral camera. LiveView has camera settings tool (based on `spectracular`), and some helpful functionalities to adjust imaging session settings. CubeView uses mainly `fpipy` and provides the view over captured cubes. Users can easily slide through raw, reflectance or radiance cubes, check radiance pixel spectra, or select frames by wavelengths to closer attention. CubeView uses `matplotlib` back-end with visualizations and saves and reads data from netCDF format.

CubeView Analyze analysis tool is created so that users can customize it according to their wishes and needs. Selected algorithms, including spectral unmixing based on vertex component analysis and fast least-squares approach, spectral angle mapper, principal component analysis and library of spectral indices are integrated to the basic version of CubeView Analyze. Users own analysis algorithms are easy to add to tool.

4.2 Algorithm

In spectral unmixing we assume that spectra are a linear mixture of some spectra, which are characteristic to the image and present in it. Now, let X be list of imaged spectra. Then linear mixture can be expressed in matrix form following way

$$X = YM, \tag{1}$$

where $M \subset R^{k\times n}$ is the mixing matrix and $Y \subset R^{d\times k}$ are characteristic spectra, which are called endmembers. Here n is number of spectra in image, d is number of spectral bands and k is number endmembers. To solve M from the equation it is necessary to determinate the endmembers Y.

There are different ways to approach to determinate Y from the imaged data. If we rely on data's geometry, we can project data to the low dimensional space and try to determinate convex hull, which is covering all or majority of data points. Now, vertices of this hull are considered as endmembers. There are several methods, which are developed to detect these vertices, such as the vertex component analysis (VCA)[6] or Pixel-Purity Index (PPI).[7] Our implemented unmixing method uses VCA to determinate endmembers.

As evident there are several ways to determinate M after endmember detection. If we discard some boundary conditions, most easiest way to solve M is to use least-square method with pseudoinversion.[8] It holds that

$$\left(Y^T Y\right)^{-1} Y^T X = M. \tag{2}$$

This quite simple matrix operation solves M for us.

Spectral angle mapper is relatively simple measurement to compare similarity of two spectra.[9] We measure angle between spectra $\mathbf{x}$ and $\mathbf{y}$ by calculating angle between these two vectors. Spectral angle

$$\theta = cos^{-1}\frac{\mathbf{x}\cdot\mathbf{y}}{\|\mathbf{x}\|\cdot\|\mathbf{y}\|}, \tag{3}$$

where $\|, \|$ is euclidean norm of the vector. By selecting some threshold for the angle θ is SAM actually a binary classificator. In analysis example user has to select reference spectra from the spectral image.

With principal component analysis (PCA) it is possible to search those components which are causing most of the variance within recorded spectra. The difference between PCA and spectral unmixing is that PCA doesn't take account any physical meaning. Thus we will just see which areas of the image is causing variance, while spectral unmixing can actually tell us, which spectra's are present in the data cube. PCA's implementation is straight forward. First, we center data around origo by extracting mean of each spectra. Then we calculate covariance matrix. At last singular value decomposition is done on the covariance matrix. As a results we will get singular values and singular vectors. Here singular values tell's us how much of the variance is explained by corresponding principal components. Now, principal components are calculated by multiplying spectra with singular numbers.

Spectral indices are ratios, sums or different arithmetic combinations of spectral bands. They return single map, which can correlate with some properties of the images object. For example, in the remote sensing of vegetation indices such as normalized difference vegetation index (NDVI) or modified chlorophyll absorption in reflectance index (MCARI). NDVI is calculated as

$$\frac{NIR - RED}{NIR + RED}, \tag{4}$$

where NIR is some near infrared band (between 0.7 to 1 μm) and RED is band from red region of the visible light (between 0.65 to 0.7 μm). MCARI is defined as

$$\frac{(R_{850} - R_{730} - 0.2\times(R_{850} - R_{570}))}{R_{730}}, \tag{5}$$

where R_i denotes to reflectance on wavelength i. Implemented indices library includes almost 300 different spectral indices.

The development focus was mainly on workflow and processes that produces magic behind the curtains: the least amount of effort is three clicks from start to whole spectral cube and one click more to start analyzing. What CubeView offers, is an effective and new way to capture cubes, show analysis results and produce quick demos to the researchers, corporate decision makers and even to consumers. It brings new innovations, wraps algorithms and scientific research in to easily approached and partly productized form.

Spectral Imaging Laboratory will release mentioned libraries and first standalone versions of CubeView during the spring 2019.

5. RESULTS

A measurement was done to test a hyperspectral imager and the new CubeView software. The selected imaging target was plastic leaves of a non-organic plant and one real leaf from a healthy plant. Without close examination both types of leaves appear to be organic to naked eye. The target was illuminated by a broad band light source to ensure that whole wavelength range of measurement is covered. The parameters of PFPI that was used in HSI for this measurement are summarized in table 2. The figure 6 shows the resulting PCA and abundance maps, as well as endmembers, which correlate to certain unique material features of the target. The measured wavelength range was $450 - 850$ nm with camera set to analog gain of 10x and exposure to 120 ms. The figure 7 shows the output of spectral angle mapper. It can be seen in figure 7, that with threshold value of 0.1 the real plant leaf is highlighted, indicating a strong similarities in spectra with the real plant. Finally, the figure 8 shows a comparison of regular RGB-composite image and a NDVI image. It can be seen, that NDVI is much higher for real plant as it is expected due to chlorophyll absorption.

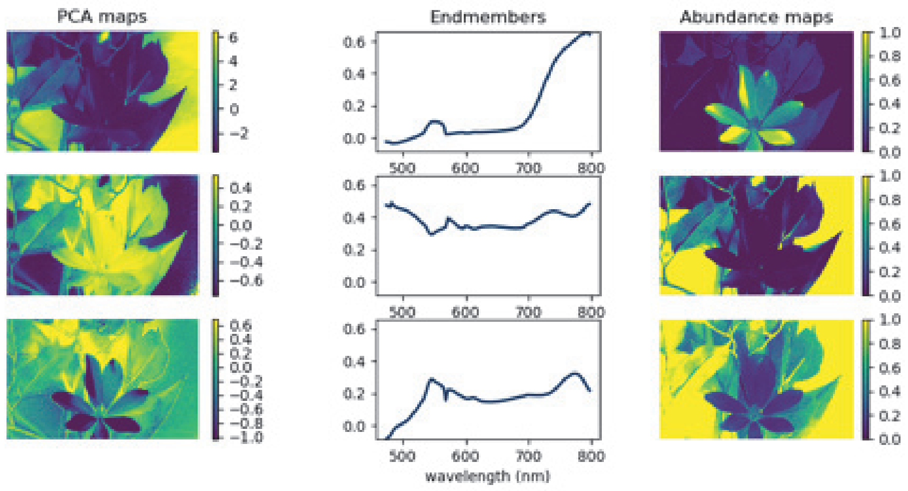


Figure 6: Results of principal component analysis (PCA) (left), and vertex component analysis (VCA) (middle and right).

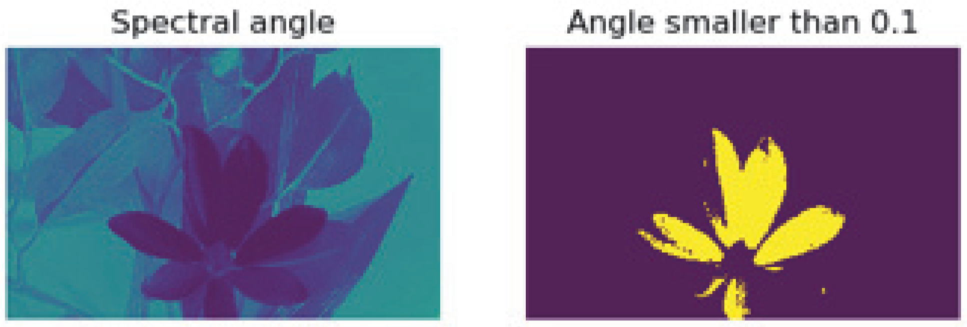


Figure 7: Spectral angle as compared to a pixel spectra from the real plant (left) and thresholded spectral angle (right).

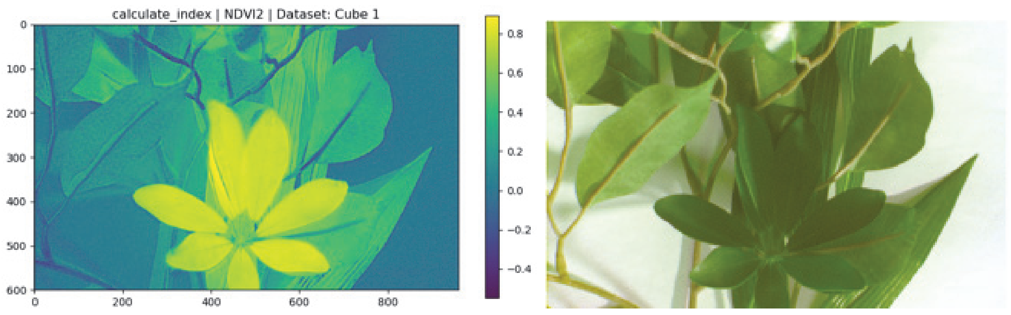


Figure 8: NDVI-index (left) and RGB-composite of three selected bands (right).

6. SUMMARY AND CONCLUSIONS

This paper presented a way to mitigate two common problems that are associated with hyperspectral imagery. The first is the unavailability of flexible and affordable imager hardware platform. The other is the raw spectral data, that are typically unusable for most of applications without an analysis algorithm. It was shown that it is possible to develop compact, low cost, portable, hyperspectral imager, that supports both MFPI and PFPI technologies. As result, the hardware platform can be used in wider range of applications, further reducing costs and increasing availability of hyperspectral imaging. In addition to the hardware, the newly developed software tool was tested with a hyperspectral camera system and it was shown, that it is possible to use PCA analysis to find and highlight the unique spectral features within raw data cube. Therefore, the user is given quick access to more visually understandable spectral information.

REFERENCES

[1] Näsilä, A., Trops, R., Stuns, I., Havia, T., Saari, H., Guo, B., Ojanen, H. J., Akujärvi, A., and Rissanen, A., "Hand-held mems hyperspectral imager for vnir mobile applications," *Proc.SPIE* **10545**, 10545 – 10545 – 9 (2018).

[2] Rissanen, A., Guo, B., Saari, H., Näsilä, A., Mannila, R., Akujärvi, A., and Ojanen, H., "Vtt's fabry-perot interferometer technologies for hyperspectral imaging and mobile sensing applications," *Proc.SPIE* **10116**, 10116 – 10116 – 12 (2017).

[3] Rissanen, A., Saari, H., Rainio, K., Stuns, I., Viherkanto, K., Holmlund, C., Näkki, I., and Ojanen, H., "Mems fpi-based smartphone hyperspectral imager," *Proc.SPIE* **9855**, 9855 – 9855 – 16 (2016).

[4] Eskelinen, M. A. and Hämäläinen, J., "Fabry-perot imaging in python." https://github.com/silmae/fpipy (2018). [Online; accessed 19-Dec-2018].

[5] Annala, L., Eskelinen, M., Hämäläinen, J., Riihinen, A., and Pölönen, I., "Practical approach for hyperspectral image processing in python," in [*International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*], **42**(3), International Society for Photogrammetry and Remote Sensing (2018).

[6] Nascimento, J. M. and Dias, J. M., "Vertex component analysis: A fast algorithm to unmix hyperspectral data," *IEEE transactions on Geoscience and Remote Sensing* **43**(4), 898–910 (2005).

[7] Chang, C.-I. and Plaza, A., "A fast iterative algorithm for implementation of pixel purity index," *IEEE Geoscience and Remote Sensing Letters* **3**(1), 63–67 (2006).

[8] Lawson, C. L. and Hanson, R. J., [*Solving least squares problems*], vol. 15, Siam (1995).

[9] Yuhas, R. H., Goetz, A. F., and Boardman, J. W., "Discrimination among semi-arid landscape endmembers using the spectral angle mapper (sam) algorithm," in [*JPL, Summaries of the Third Annual JPL Airborne Geoscience Workshop*], **1**, 147–149 (1992).

PV

DANGERS OF DEMOSAICING: CONFUSION FROM CORRELATION

by

Matti Eskelinen, Jyri Hämäläinen 2019

WHISPERS 2018 : 9th Workshop on Hyperspectral Image and Signal Processing, Evolution in Remote Sensing. IEEE. DOI:10.1109/WHISPERS.2018.8747204

DANGERS OF DEMOSAICING: CONFUSION FROM CORRELATION

Matti A. Eskelinen, Jyri Hämäläinen

University of Jyväskylä
Faculty of Information Technology

ABSTRACT

Images from colour sensors using Bayer filter arrays require demosaicing before viewing or further analysis. Advanced demosaicing methods use empirical knowledge of inter-channel correlations to reduce interpolation artefacts in the resulting images. These inter-channel correlations are however different for standard RGB cameras and hyperspectral imagers using colour sensors with added narrow-band spectral filtering.

We study the effects of conventional demosaicing methods on hyperspectral images with a dataset originally collected without a colour filter array. We find that using advanced methods instead of bilinear interpolation results in an overall increase of 9–14 % in absolute error and a decrease of 1–3 % in PSNR, but also observed a decrease in MSE of 11–13 %.

For the corresponding RGB images, the advanced methods improved fidelity as expected. The results also demonstrate that the reconstruction methods that take advantage of correlation transport noise present in a single component to other reconstructed layers.

Index Terms— Hyperspectral imaging, Fabry-Perot, Instruments, Bayer pattern, Colour filter array (CFA) interpolation, Demosaicing, Algorithms

1. INTRODUCTION

Modern hyperspectral “snapshot” imagers based on Fabry-Perot interferometers as their filtering technique can use either monochromatic or Bayer-filtered sensors as their imaging component. The latter case allows imaging of multiple narrow wavebands with a single exposure, as the radiances for each single wavelength can still be reconstructed based on the known quantum efficiencies of the different pixel filters for the given wavelengths [1]. However, since only a measurement of a single colour filter is available at each location, interpolation (or *demosaicing*) has to be used to approximate the other filter responses at each location in order to solve for spectral radiances.

This research is in part funded by the Jane and Aatos Erkko foundation (grant 170015) and in part by the Finnish Funding Agency for Innovation (TEKES) (grant 1711/31/2016).

Much work has been put into the development of demosaicing methods due to the prevalence of Bayer filters in digital consumer cameras. Reducing interpolation artefacts without sacrificing too much computational efficiency is a problem that has provided us with a wealth of existing literature. Many approaches are based on utilising the correlation of the luminosity and chromaticity in natural images to improve the SNR of the reconstruction [2, 3]. We give a short summary of the methods under comparison in section 2.

The correlations are partially reliant on the fact that the colour filters of the typical Bayer filter array are approximating human vision and thus have overlapping transmittances across a range of wavelengths. However, for hyperspectral imagers using the Bayer filter reconstruction technique with narrow-band filtering, the correlation between the signals of different colour filters is dependent on the wavelengths transmitted by the narrow-band filter. While demosaicing methods have been developed also for imagers with multispectral filter configurations [4], there is little literature on the applicability of existing methods in use cases related to spectral imaging.

To study the effect this discrepancy has on the resulting data, we construct a benchmark for the radiance reconstruction using radiances from data taken using a more traditional filter system. The procedures used are detailed in section 3. We use the benchmark to compare the more advanced methods against bilinear filtering – which does not rely on correlations – and present our findings in section 4.

2. DEMOSAICING METHODS

A CFA image is a matrix composed of three (or more) different kinds of pixels in a given pattern. An example of a typical Bayer “RGGB” pattern is given in figure 1, with R, G and B referring to the red, green and blue filters respectively. Demosaicing entails the computation of a full-size layer of values for each of the component types, with the missing values in other locations filled in using interpolation.

We compared three demosaicing methods that had readily available software implementations, specifically those implemented by the “Colour - demosaicing” Python library [5]. What follows is a brief summary of each, for the readers disinclined to wade through the references.

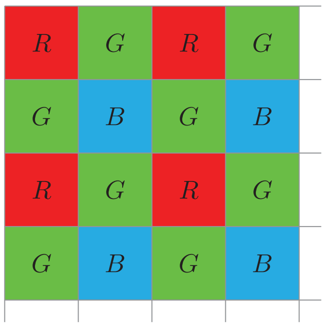


Fig. 1. Example of an RGGB Bayer filter pattern.

2.1. Bilinear interpolation

Bilinear interpolation fills each missing value with the average of its nearest neighbours. Interpolation is done on each component plane independently, without any assumptions of correlations between the components. While it can be implemented highly efficiently using a discrete convolution, it causes highly visible artefacts in the resulting image. [6]

2.2. Malvar (2004)

The method of Malvar, He and Cutler [2] is similar to the simple bilinear interpolation, but adds to each interpolated value a correction based on the estimated gradients of the other components, depending on the location of the pixel with respect to the pattern. Like bilinear interpolation, it is also implemented as a linear convolution, making it computationally efficient. It is familiar to many as the default demosaicing method in the MATLAB Image Processing Toolbox [7].

The precise weights used for the convolution implementation of their algorithm are the result of an experimental fit to find the best reconstruction of the Kodak colour image dataset, with a constraint on the divisibility to find an efficient binary representation.

2.3. Menon (2007)

The method of Menon, Adriani and Calvagno [3] utilizes directional filtering and *a posteriori* selection of the best interpolation of the green component. The other colour components in the green locations are reconstructed using bilinear interpolation.

The directional information in the green layer is then used to help reconstruct the colour difference layers $(R - B)$ and $(B - G)$, which are then used to recreate the red and blue components at other locations, along with corrections based on frequency filtering. The approach relies heavily on the correlation between the channels, and is roughly 3-5 times more computationally intensive then the bilinear interpolation.

3. TEST METHODOLOGY

3.1. Radiance dataset

In order to study the effect of demosaicing on hyperspectral images we needed data with a similar structure to the output of Fabry-Perot imagers, but which has been collected without a Bayer filter. The data could then be converted to a mosaic trough simple omission and the results of demosaicing could be reliably compared to the original without having to worry about any systematic error arising from previous reconstruction methods.

We decided to use as the base data the publicly available radiance images of natural scenes by Foster, Nascimento and Amano [8], which were acquired using a full-frame imager with a tunable filter and a monochromatic (non-array) sensor. The characteristics of the natural scenes also resemble those of the Kodak dataset often used in demosaicing studies, with varying edge features and spectral signatures. Each of the thirty images had 33 radiance bands evenly distributed between wavelengths of $400\,\mathrm{nm}$ and $720\,\mathrm{nm}$ with a band width of $10\,\mathrm{nm}$.

The dataset also included an RGB image of each scene reconstructed from the radiance using the CIE standard observers. These were used as-is to test the performance of the methods for this dataset.

3.2. Mosaic construction

For each of the 33 band radiance cubes in the dataset, we constructed a corresponding mosaic cube of 11 CFA images with an RGGB pattern with each image having its R, G and B pixels sampled from different wavelength bands of the cube. The wavelengths used for each mosaic layer are listed in table 1. The choice of the specific pattern was arbitrary. While we did not verify it experimentally, the different orderings of the colour components should produce equivalent results since none of our fidelity measures use directional quantities which might change under different patterns.

The wavelengths were selected by simply dividing the set of radiance bands into three equal parts, and using the longer wavelengths for the red, the medium for green and shorter for blue in consecutive triplets, such that no triplet contained bands near each other in wavelength. Since there is no sensor-induced correlation in the different radiance values, the ordering of the bands should not matter; We opted to match the wavelengths roughly to the colour components they affect the most. This way any correlations between the radiances present in the natural scenes due to spectral properties of the specific scene should be roughly similar to those in true-colour images.

This method of construction approximates the structure of data from a spectral imager using a scanning Fabry-Perot interferometer with three of the transmission peaks in the range

Table 1. Wavelength bands from of the base image used for the different pixels in each mosaic layer of the corresponding test dataset.

#	λ_B	λ_G	λ_R
1	400	510	620
2	410	520	630
3	420	530	640
4	430	540	650
5	440	550	660
6	450	560	670
7	460	570	680
8	470	580	690
9	480	590	700
10	490	600	710
11	500	610	720

of sensor sensitivity at once. It is simplified in that the different transmission peaks are not mixed at all in the R, G and B pixels unlike in a real imager, where the response of each pixel to the peaks would be dependent on the wavelength.

We decided against the modelling of a more realistic situation for this study, since the choices needed for the construction of the more realistic mosaics would complicate interpretation of any results and proper analysis would have taken more time then was available for the preparation of this paper.

3.3. Fidelity criteria

Since perceptual metrics based on human vision do not make sense for the radiance data directly, we consider only objective fidelity measures based on the numerical difference of the original and the reconstructed image. Following [6], we measure the fidelity of the demosaiced images with $X \times Y$ pixels for each band b using the mean absolute error,

$$\mathrm{MAE}_b(I, I') = \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \left| I_{x,y} - I'_{x,y} \right|$$

mean square error,

$$\mathrm{MSE}_b(I, I') = \frac{1}{XY} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \left(I_{x,y} - I'_{x,y} \right)^2$$

and peak signal-to-noise ratio

$$\mathrm{PSNR}_b(I, I') = 10 \cdot \log_{10} \left(\frac{\max(I_b)^2}{\mathrm{MSE}_b(I, I')} \right)$$

with the original radiance images I and the reconstructions I'. The same metrics were also calculated for the RGB images and their reconstructions using colour components in place of radiance bands.

Table 2. Average fidelities of the Malvar and Menon methods relative to bilinear filtering on the radiance and RGB images.

	Radiance		RGB	
	Malvar	Menon	Malvar	Menon
MAE	1.09	1.14	0.97	1.15
MSE	0.79	0.77	0.54	0.66
PSNR	0.99	0.97	1.15	1.11

The three fidelity measures were averaged over the dataset for each band and ordered to match the structure of the original radiance data. The metrics for each method were also divided by the corresponding metric for bilinear interpolation to get a touch on their relative performance. Corresponding values were also calculated for the RGB reconstructions to verify that the methods perform as expected for the kind of images they were designed for.

4. RESULTS

Figures 2, 3 and 4 show the MAE, MSE and PSNR for the different methods. The metrics were calculated for each wavelength band and averaged over the test dataset. The absolute values show a significantly worse reconstruction for the first band in all the cases. This is due to the high noisiness of the data in the 400nm band, which was known for the dataset and confirmed by us with a visual inspection. This noisiness carries over to the relative metrics for the Malvar and Menon methods, in that the reconstruction of 510nm and 620nm bands included in the same mosaics with the noisy waveband suffer with the methods using correlation. This was also visually apparent in the reconstruction of the given bands.

The relative fidelities of the Malvar and Menon methods over all the bands are presented in table 2 for both the radiance and RGB reconstructions. For comparison, Malvar, He and Cutler report an increase of 5.68 dB in PSNR for their method over the bilinear interpolation [2], while Menon, Adriani and Calvagno claim an increase of 9.69 dB [3]. Their tests were conducted respectively on sets of 15 and 20 RGB colour images from the Kodak dataset.

5. CONCLUSIONS AND FUTURE WORK

The relative fidelity measures demonstrate that introducing a correlation-based correction in the interpolation also has the side effect of carrying over any noise present in the bands to the other interpolated layers, which is to be expected. Similar effects are present in the reconstruction of radiance measurements from actual Fabry-Perot hyperspectral imagers in the bands where correlation is introduced by the mixing of the different narrow-band radiances on the sensors (though dif-

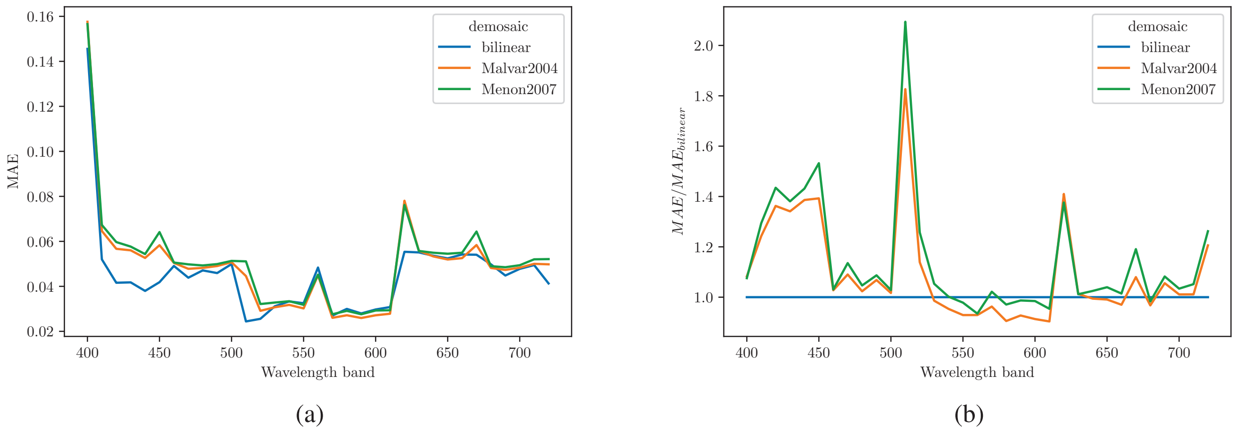


Fig. 2. (a) Average MAE over the full dataset for each wavelength band. (b) Average MAE compared to the bilinear interpolation.

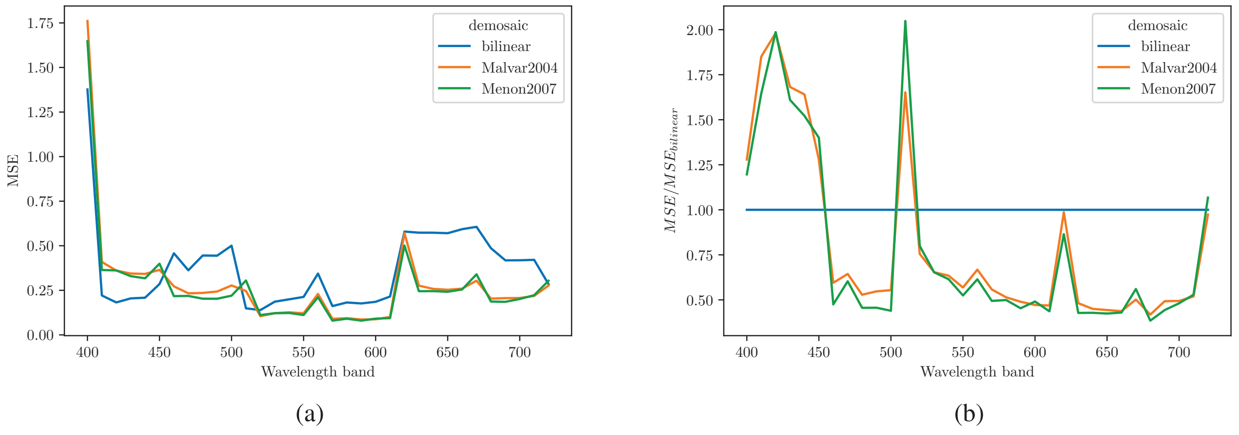


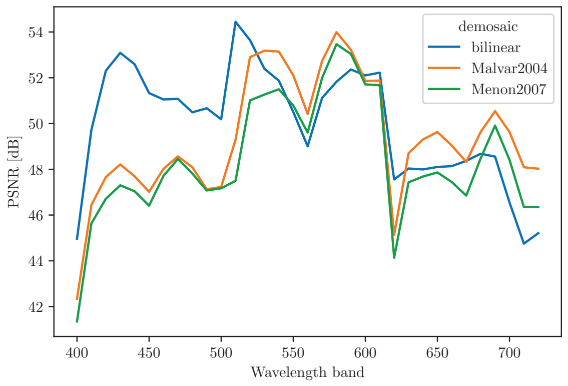
Fig. 3. (a) Average MSE over the full dataset for each wavelength band. (b) Average MSE compared to the bilinear interpolation.

ferent in magnitude to the usual correlation of the R, G and B pixels).

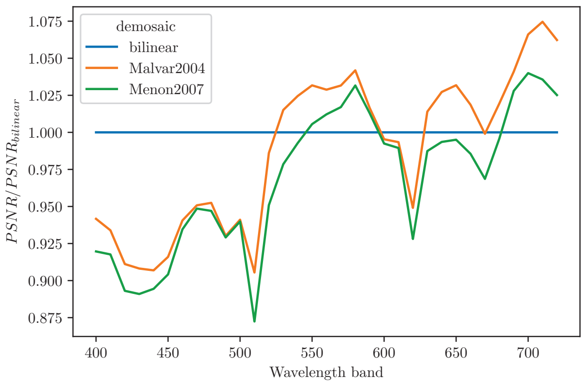
All the measures also show generally worse performance for the Malvar and Menon methods in the noisy band, which is most likely due to their use of gradients (which are generally sensitive to noise). However, the advanced methods perform slightly better in the bands with lower noise. While we did not comprehensively study the effect of the ordering of the bands, some tests ran after the calculation of the main results suggest that the ordering does not significantly change the overall performance, although some differences arise on the single bands. This suggests that the methods are not very sensitive to the chromatic correlations in the natural scenes, but that differences arise based on the exact band used for the better sampled component (usually the green one).

The relative fidelities in table 2 show that the overall fidelity of the advanced reconstruction is decreased by a few percent compared to the bilinear interpolation. The results on reconstructed RGB images show that when the assumptions of the algorithms are fulfilled, the fidelity of the reconstruction should increase significantly for this specific dataset.

While we did not evaluate any perceptual metrics, visual inspection of some of the reconstructions suggested that while the advanced methods generally did not cause noticeable degradation in the radiance images, localised artefacts were generated for certain materials (spectral signatures) in the scenes. More comprehensive study comparing spectral differences using e.g. spectral angle or correlation would be needed to quantify these effects.



(a)



(b)

Fig. 4. (a) Average PSNR over the full dataset for each wavelength band. (b) Average PSNR compared to the bilinear interpolation.

6. REFERENCES

[1] Heikki Saari, Ilkka Pölönen, Heikki Salo, Eija Honkavaara, Teemu Hakala, Christer Holmlund, Jussi Mäkynen, Rami Mannila, Tapani Antila, and Altti Akujärvi, “Miniaturized hyperspectral imager calibration and UAV flight campaigns,” in *Proc. SPIE*. Oct. 2013, vol. 8889, p. 88891O, International Society for Optics and Photonics.

[2] Henrique S Malvar, Li-Wei He, Ross Cutler, and One Microsoft Way, “High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images,” in *International Conference of Acoustic, Speech and Signal Processing*. may 2004, pp. 5–8, Institute of Electrical and Electronics Engineers, Inc.

[3] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno, “Demosaicing With Directional Filtering and a posteriori Decision,” *IEEE Transactions on Image Processing*, vol. 16, no. 1, pp. 132–141, jan 2007.

[4] Pierre-Jean Lapray, Xingbo Wang, Jean-Baptiste Thomas, and Pierre Gouton, “Multispectral Filter Arrays: Recent Advances and Practical Implementation,” *Sensors*, vol. 14, no. 11, pp. 21626–21659, Nov. 2014.

[5] The Colour Developers, “Colour - Demosaicing,” Online, 2018, Software available at http://colour-science.org/colour-demosaicing/.

[6] O. Losson, L. Macaire, and Y. Yang, “Comparison of Color Demosaicing Methods,” in *Advances in Imaging and Electron Physics*, vol. 162, pp. 173–265. 2010.

[7] Mathworks Nordic, “Convert Bayer pattern encoded image to truecolor image - MATLAB demosaic,” Online, https://se.mathworks.com/help/images/ref/demosaic.html.

[8] Sérgio M. C. Nascimento, Kinjiro Amano, and David H. Foster, “Spatial distributions of local illumination color in natural scenes,” *Vision Research*, vol. 120, pp. 39–44, Mar. 2016.