https://doi.org/10.35662/unine-thesis-2727

Short-wave infrared hyperspectral imaging of minerals

thesis presented at the

Faculty of Sciences

Center for Hydrogeology and Geothermics
(CHYN)

University of Neuchâtel, Switzerland

for the degree of

Doctor of sciences

presented by

Laurent Fasnacht

accepted on the recommendation of

Prof. Philip Brunner
Prof. ass. Philippe Renard
Prof. Thomas Südmeyer
Prof. Grégoire Mariéthoz
Dr. habil. Jens Becker

Defended on December 12^{th}, 2018

UNIVERSITÉ DE NEUCHÂTEL

Faculté des Sciences
Secrétariat-décanat de Faculté
Rue Emile-Argand 11
2000 Neuchâtel – Suisse
Tél : + 41 (0)32 718 21 00
E-mail : secretariat.sciences@unine.ch

IMPRIMATUR POUR THESE DE DOCTORAT

La Faculté des sciences de l'Université de Neuchâtel autorise l'impression de la présente thèse soutenue par

Monsieur Laurent FASNACHT

Titre:

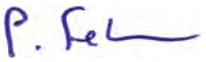
"Short-wave infrared hyperspectral imaging of minerals"

sur le rapport des membres du jury composé comme suit:

- Prof. Philip Brunner, directeur de thèse, UniNE
- Prof. ass. Philippe Renard, co-directeur de thèse, UniNE
- Prof. Thomas Südmeyer, UniNE
- Prof. Grégoire Mariéthoz, Université de Lausanne
- Dr. habil. J. Becker, NAGRA, Wettingen

Neuchâtel, le 11 janvier 2019

Le Doyen, Prof. P. Felber



Acknowledgements

First, I would like to heartfully thank Philip Brunner for supervising my thesis, and Philippe Renard for his co-supervision. Their enthusiasm, trust, and academic experience were invaluable for this thesis.

I would like to thank the Swiss National Cooperative for the Disposal of Radioactive Waste (NAGRA) for participating in the funding of this thesis, and Jens Becker for his support and advice. The practical applications proposed, and the problems that arose, were a key factor motivating the work presented.

I am also deeply grateful to my co-workers, for all their help and feedback that they provided. In particular, I have greatly benefited from the expertise in mineralogy of Marie-Louise Vogt. I'd like also to thank Laurent Marguet, who helped me start by showing me how the hardware was supposed to work, and helped a lot for practical and field work.

I finally want to thank my family for their support all along my studies. My parents deserve a special thank for sharing their interest in knowledge, more precisely in the value of engineering, science and language, which were key factors for the success of this thesis.

Special thanks to Solange Oesch for her love and support, and to my friends who distracted me just enough to make my thesis successful.

Abstract

Hyperspectral imaging is a promising technology for mineral identification and mapping. This technology is commonly used for remote sensing, using aerial sensors. It is however less frequently used for laboratory measurement, as a range of issues currently undermine the quality of the data obtained, and consequently the resulting conclusions. Mineral identification consists in two steps: 1. data acquisition and calibration, 2. classification, using a spectral library as reference data. This thesis presents an analysis of this whole process and proposes improvements.

The camera was carefully studied in order to understand exactly its operating principles. Each aspect of the scene set-up, such as the type of light source, the positioning of the various elements of the set-up and the impact of surrounding objects was carefully evaluated. Algorithms were designed to improve the efficiency of the acquisition, such as automate focus or to avoid difficulties in choosing the correct exposition. Methods to combine multiple images in order to increase the data quality were also developed: high dynamic range was implemented to merge multiple images captured with different integration time, and a method was developed to separate the transmittance and the reflectance of translucent objects.

The efficiency of artificial neural networks was evaluated for classification. It was found that they were producing highly accurate results, but required complete spectra as input, which requires interpolation of bad pixels and prevents the re-use of an already trained network with a different camera. Therefore a novel input layer was designed which handles efficiently spectra with missing bands. As no training data was readily available for training an artificial neural network for mineral identification, a spectral library of 2D reflectance images of 130 samples of 76 distinct minerals was created. To the best of our knowledge, no similar dataset exists.

Finally, hardware and software tools were designed and implemented to carry out all these steps and to allow efficient automation of all the devices involved. These tools have been published under Open Source licenses.

Keywords

hyperspectral; calibration; laboratory; illumination; dark frame; non-uniformity correction; focus; high dynamic range; HDR; translucent; infrared; short-wave infrared; neural network; multi-layer perceptron; input layer; Haar transform; bad pixel; missing bands; spectral library; minerals

Résumé

L'imagerie hyperspectrale est une technique prometteuse pour l'identification et la cartographie de minéraux. Elle est largement employée en télédétection, au moyen de capteurs aéroportés, mais est relativement peu utilisée en laboratoire. De fait, un ensemble de problèmes limite la qualité des données obtenues, et par conséquent celle des conclusions en découlant. L'identification de minéraux nécessite deux étapes: 1. acquisition des données et calibration, 2. classification, en utilisant une librairie spectrale en tant que données de références. Cette thèse présente une analyse de l'entier de ce processus et propose des améliorations.

La caméra a été soigneusement étudiée afin de comprendre son principe de fonctionnement. Les différents aspects de l'acquisition ont été soigneusement évalués, notamment le choix du type de source de lumière, la disposition des différents éléments lors de l'acquisition de données, ainsi que l'impact des éléments environnants. Des algorithmes ont été conçus afin d'améliorer l'efficacité de l'acquisition de données, notamment pour automatiser le focus ou pour éviter les difficultés à choisir l'exposition correcte. Des méthodes combinant plusieurs images afin d'améliorer la qualité ont aussi été développées: de l'imagerie à grande gamme dynamique (HDR) a été implémentée afin de combiner plusieurs images capturées avec des temps d'intégration différents, et une méthode a été développée pour séparer la transmittance et la réflectance d'objets translucides.

L'efficacité des réseaux de neurones artificiels a été évaluée pour la classification: ceux-ci produisent des résultats très précis, mais requièrent un spectre complet. Cette exigence nécessite l'interpolation de pixels défectueux et empêche la réutilisation d'un réseau déjà entraîné avec une autre caméra. Ainsi, une nouvelle couche d'entrée (input layer) a été conçue, laquelle gère de façon efficace des spectres dont certaines bandes manquent. Comme il n'existait pas de données d'entraînement à l'identification de minéraux d'un tel réseau de neurones, une librairie spectrale d'images de réflectance a été créée. Celle-ci contient 130 échantillons de 76 minéraux distincts. Une telle librairie est, à notre connaissance, unique.

Finalement, des outils logiciels et matériels ont été développés afin d'effectuer toutes ces opérations de façon efficace, et d'automatiser tous les équipements concernés. Ces outils ont été publiés sous licence Open Source.

Mots clés

Hyperspectral; calibration; laboratoire; illumination; trame noire; correction de non-uniformité; focus; imagerie à grande gamme dynamique; HDR; translucide; infrarouge; infrarouge ondes courtes; réseau neuronal; perceptron multi-couche; couche d'entrée; transformée de Haar; pixel défectueux; bandes manquantes; librarie spectrale; minéraux

Contents

Chapter 1

Introduction

The work presented in this thesis deals with mineral identification using Shortwave Infrared (SWIR) hyperspectral imaging. Mineral identification is of key interest for many applications, and using remote sensing technique has multiple advantages. First, the results are obtained very fast, and it generates directly a map, instead of needing multiple point-wise analyses. Additionally, identifying mineral using these techniques has the advantage of not affecting rock integrity, which would be the case if samples were collected for laboratory measurement. These key properties are interesting for example for the Swiss National Cooperative for the Disposal of Radioactive Waste (NAGRA).

A short introduction about the concepts required, remote sensing (section 1.1) and spectral data processing (section 1.2), is given hereafter. Additional information is also given in the introduction to the articles.

1.1 Remote sensing

Remote sensing is the acquisition of information about an object (from a small sample to a large land area), without physical contact. It generally consists in electromagnetical measurement from airborne devices, such as drones, planes or satellites. One of the main characteristics of interest of a remote sensing device is the range of the electromagnetical spectrum it operates on. Some devices operate using radio waves (such as radars and radiometers), other using optical radiations (such as lidars and spectrometers). Optical radiations are usually subdivided into spectral ranges [1] (fig. 1.1). Sensors measure the radiance. Radiance is the power emitted per surface and per solid angle. Intuitively, it consists in the amount of light emitted by a surface as "perceived" by the sensor.

There are two main types of remote sensing, active and passive. Active

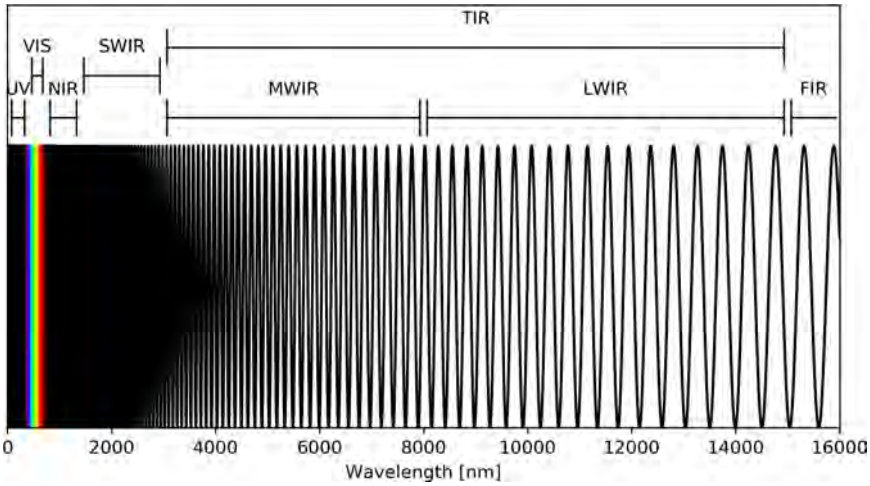


Figure 1.1: Commonly used spectral range names in the optical radiation range. UV: ultraviolet, VIS: visible (light), NIR: near infrared, SWIR: short wave infrared, MWIR: medium wave infrared, LWIR: long wave infrared, FIR: Far infrared, TIR: Thermal infrared.

remote sensing is where a signal is sent to the object, and the "response" is observed, while passive remote sensing is only observation [2]. Common examples of active remote sensing is lidar and radar. Passive remote sensing examples are any imaging device, radiometer and spectrometer, such as satellite imagery or airbone photography. Remote sensing devices referred to as "imaging" produce a multi-dimensional array of values, which can be seen as an image (which in most of the cases does not correspond to what would be seen by a human eye). For example, a spectrometer will provide the spectrum at one point, whereas an imaging spectrometer will provide a 2D array containing spectral information for each point. Imaging devices often are line scan ("push-broom") devices. They capture one line of pixels at a time, and use the movement of the sensor to create a 2D image [2] (fig. 1.2). Devices that directly capture a 2D image are called full-frame devices, or area-scan devices.

Usually, the field of view of the camera is fixed for a given camera and lens, so the level of details depends mostly on the sensor characterics and the distance between the camera and the sample. An important characteristic of an imaging device is the spatial resolution, which indicates the size of a pixel on the sample. For example, an image could have a $4\,\mathrm{mm}^2$ per pixel spatial resolution, which means that every pixel in the image will correspond to light coming from a $4\,\mathrm{mm}^2$ area. If the spatial resolution is low, each pixel will cover a large area and may be a mix of multiple different values

(like a blurry image). On the other hand, a very high spatial resolution will require multiple captures on the same sample because the area covered by each acquisition will be very small.

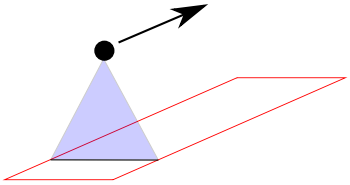


Figure 1.2: Illustration of a push-broom sensor. The sensor captures a single line, and the movement of the sensor allows the capture of a stripe (in red), resulting in a 2D image.

We will focus on imaging devices operating on optical radiations. For these devices, the term "band" corresponds to a spectral range that is captured as a value. In addition, two terms are commonly used, despite being mostly marketing words: multispectral cameras and hyperspectral cameras. A multispectral camera is an imaging device which captures more than one band, and whose bands are usually quite wide and separated. Hyperspectral cameras, on the other hand, have many contiguous bands. An illustration of the difference between hyperspectral and multispectral is shown in figure 1.3. These devices measure for each spatial pixel and each band a number proportional to the spectral radiance of the target at that position in the wavelength range corresponding to the band. These values are usually written with a unit of DN (digital number).

The camera used in this thesis is a push-broom SWIR hyperspectral camera. The wavelengths referred to by the term "short wave infrared" vary according on the author. In the present case, the camera captures wavelengths between 937 nm and 2539 nm (more details in section 5.1.1). SWIR has the advantage of being usually reflected (unlike thermal infrared or far infrared), of being transmitted through glass and of being far enough from visible light in order to provide observations of features differing from the visible and near infrared features. A disadvantage of SWIR is that sensors are more expensive, and are more sensitive to noise. Sensors like the ones used in cameras operating on visible light (silicon-based sensor) are not sensitive to wavelengths above 1000 nm.

In this thesis, we focus on laboratory measurements, which share similarities with airborne remote sensing. However, since we can choose the setup (like illumination type, camera position and sample position), better contrasts are usually achievable. On the other hand, there are additional difficulties, for example related to the geometry of the scene, the focus or the surrounding objects.

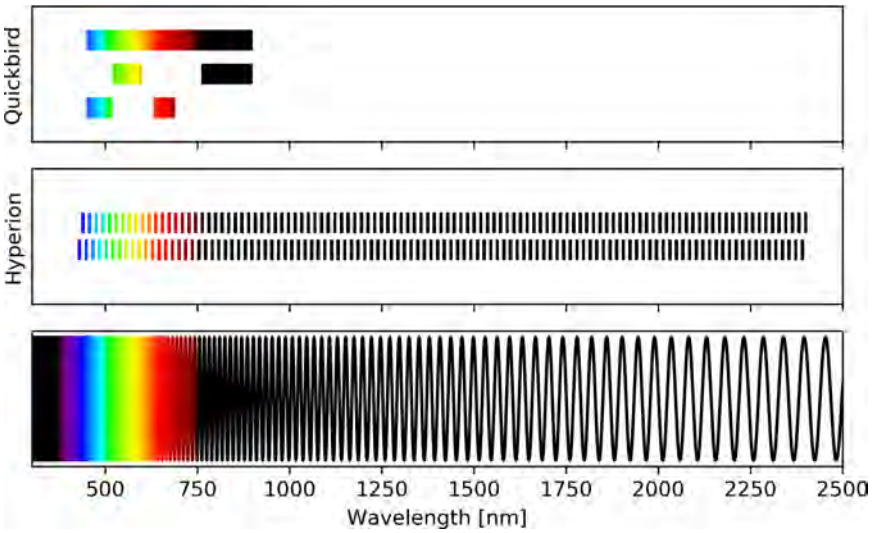


Figure 1.3: Comparison between a multispectral and hyperspectral imaging devices. Quickbird has 5 bands: Panchromatic (450-900nm), Blue (450-520nm), Green (520-600nm), Red (630-690nm), NIR (760-900nm), and is therefore considered to be multispectral. EO-1 Hyperion is considered hyperspectral, as it has 220 bands from 430 to 2400nm.

In general, we are interested in physical properties of the target, such as its transmittance or reflectance. Reflectance is the proportion of light reflected by the sample, while the transmittance is the proportion transmitted (fig. 1.4). Usually, we compute the reflectance and transmittance per band, meaning that we have reflectance and transmittance spectra. Intuitively, reflectance corresponds to the color of the object, while transmittance corresponds to the color of light going through an object. As measuring the incoming light is non-trivial, the method usually applied is to use a white reference as a comparison point, whose reflectance is assumed to be 1 for every band. Therefore, we can use as reflectance the ratio between the light reflected by the sample and the light reflected by the white reference.

However, issues like non-uniform illumination, backscattering on surrounding objects, sensor non-uniformity or sensor non-linearity may affect the reflectance measurement, as the incoming light estimated using the white reference and the effective light illuminating the sample may not be the same. Data acquisition set-ups should be throughoutly thought out because they are very sensitive to such issues.

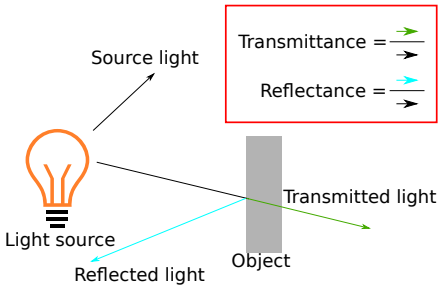


Figure 1.4: Illustration of the reflectance and transmittance concepts.

1.2 Data processing

In general, reflectance data itself is not the goal of multispectral and hyperspectral measurement, the aim is to identify characteristics of the target. Indices are commonly used for multispectral imaging, due to the low number of bands. They consist in a function of the band values, usually with some kind of ratio. Numerous indices exist. As an example, perhaps the most well-known indice is the Normalized Difference Vegetation Index (NDVI) [3], which uses only the red and infrared band to estimate whether a pixel contains green vegetation or not. This kind of approach is usually not applied to hyperspectral data, as the large number of bands allow more precise approaches.

1.2.1 Dimensionality reduction

Hyperspectral data can be seen as data in a multidimensional vector space, with each pixel being a vector in that space. Practically, this means that every pixel of the image can be represented as a tuple of numbers, each one being the measurement of one specific band. For example, if the hyperspectral camera has 256 bands, each pixel can be seen as a point in $\mathbb{R}_{+}^{256}$ (fig. 1.5). Most classification algorithms work in that space, since they consider the spectra. It may seem a straightforward problem, however working with high dimensionality space creates additional challenges, because the volume of the space increases very quickly as the number of dimensions increases, which makes all the data points very sparse in the space. Sparsity creates problems for sampling and finding neighbors.

One of the classical approaches to work around the problems related to high dimensionality is dimensionality reduction. The intuition of dimensionality reduction is that the different bands are strongly correlated, and that a given spectrum can be expressed with a limited combination of base spectra (fig. 1.6). There are numerous ways of reducing dimensionality.

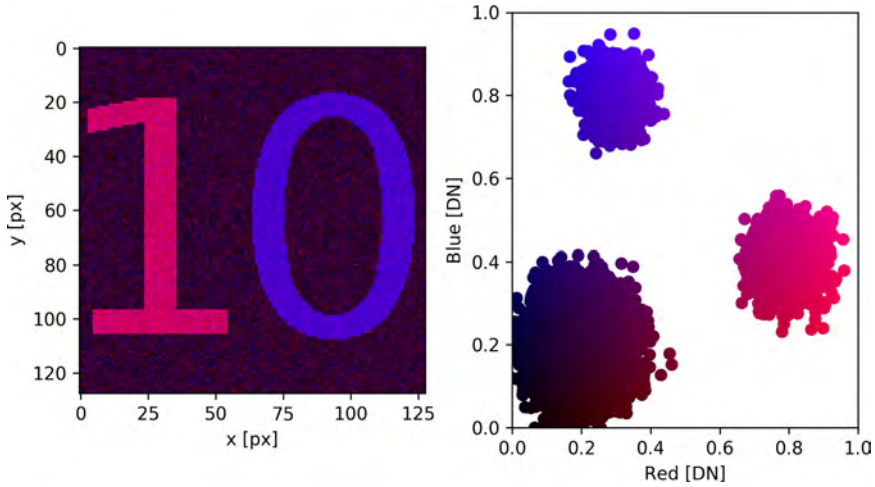


Figure 1.5: Illustration of spectral space. The original 2-band image (on the left), each spatial pixel is represented by its color. On the image on the left, the pixels are positioned in a 2D dimensional space, depending only on the values of the bands. The concept is similar for images with a higher number of bands, but it is difficult to represent higher dimensionality space in an intuitive way.

The most common ones are principal component analysis (PCA) [4] and maximum noise fraction[1] (MNF) [5]. Extensions and other types exist such as standardized PCA [6], generalized MNF [7] and numerous others [8].

PCA consists in finding a orthogonal basis of the spectral vector space (the components), in such a way that the data projected onto the space generated by each individual component is in decreasing order of variances. An example of PCA is depicted in figure 1.7. MNF works in a similar way, but orders the components in decreasing signal-to-noise ratio. These methods are very powerful when used on hyperspectral images, but the components found have very little physical sense, and they are not the "nice curves" that are shown in figure 1.6. Another drawback of these methods is that the transformation obtained is dependant on the data used to estimate it, which may lead to problems if new data is available afterward.

1.2.2 Classification

One approach that can be applied either directly to spectral data or after dimensionality reduction is classification. There are two big types of classi-

[1] also called minimum noise fraction

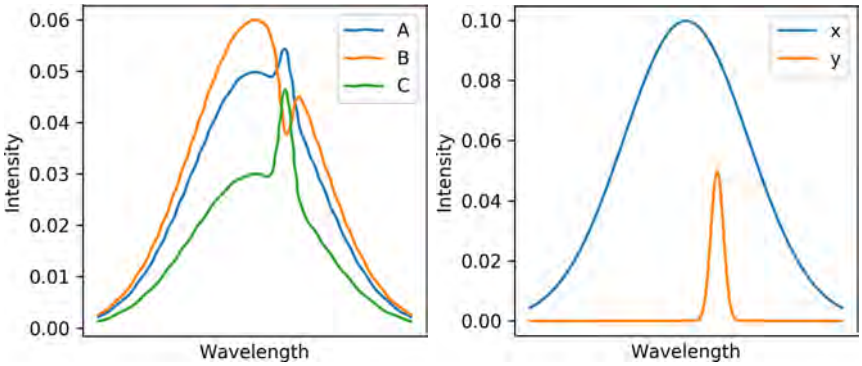


Figure 1.6: Illustration of a dimensionality reduction. On the left, 3 synthetic "measured" spectrum (A, B and C). One can see that they are composed of a linear combination of the two spectra x, y on the right plot, and some noise. Dimensionality reduction will return $A \approx 0.5x + 0.2y$, $B \approx 0.6x - 0.3y$, $C \approx 0.3x + 0.4y$.

fication: unsupervised and supervised methods. Unsupervised methods are basically clustering, their aim is to find set of points in the spectral that are of similar spectrum. Supervised methods, on the other hand, use reference data (spectral library) to make the classification, they assign measured spectra to one of the classes of the reference data. Methods differ mostly by the definition "similar" spectra, and what hypotheses on the class are expected (e.g. the number of class and their topology).

A very basic example of clustering is the k-means method [9]. The number of expected classes k has to be provided. It consists in finding the k points, the "center" of the classes, where the sum of the distance of the points to their class center is minimal. It starts by selecting a spectrum as a center for each of the classes. Each update steps consists in finding for each point the closest class center, and assign this point to the class. Then the centers are updated by computing the average of the class. An example is shown in figure 1.8. Even if the method is unsupervised, parameters such as the choice of the initial points and the number of class have to be given. The result of the algorithm vary depending on the parameters.

Supervised methods, on the other hand, use reference data. One of the simplest possible methods, and perhaps one of the most commonly used, is the spectral angle mapper [10]. It associates points to the closest reference spectra in terms of angle in the spectral space, as long as the angle is smaller than a threshold.

As these methods are applied pixel-wise, it is often preferable to add spatial constraints. This has been done for example for k-means [11].

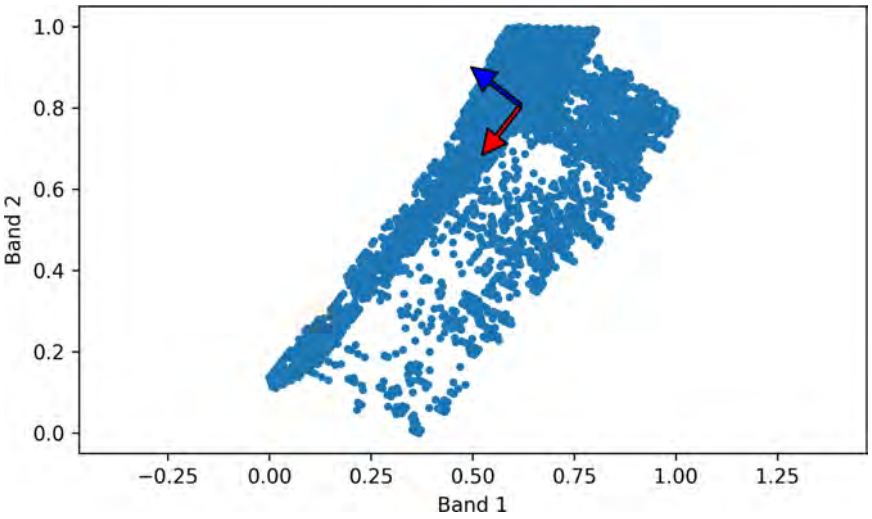


Figure 1.7: PCA example on a 2 band image. Each pixel is represented by a point. The red arrow represents the first component, and the blue one the second one. We can see that the first component is oriented in the direction which has the most variance, and the second one is orthogonal to the first one. The coordinates of each point can be represented using the basis generated by these two vectors, instead of its values for each band.

1.2.3 Unmixing

Instead of assigning a class to each pixel, the pixel measurement can also be considered as resulting of a mix of reference spectra. The goal of an unmixing method is to decompose a given spectrum as a combination of classes. Unmixing methods can be either based on a physics, or be data driven. The main difference between methods is the mixing model used (i.e. how the spectrum measured is expected to be produced from the reference spectra). Due to the number of different methods available, it is out of the scope of this introduction to do a full review. An excellent review of these methods is available [12].

A very common physics-based model in remote sensing is the linear mixing model [13]. It states that the measured spectra is a linear combination of the reference spectra. This model is valid if the spectrum is optically mixed (a spatial pixel covers a mix of multiple “pure” areas), but is not suitable for mixtures or samples which are not flat. In aerial remote sensing, bilinear models can be used if we assume that the light has at most two reflections.

More advanced models exists, based on nonlinear mixing models. They

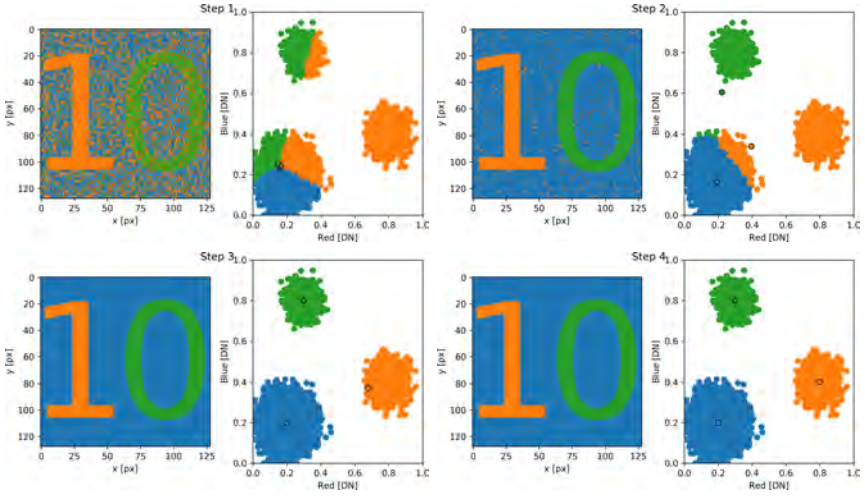


Figure 1.8: Illustration of the k-means method. Colors correspond to the class. The points circled in black are the centers.

provide much improvement for intimate mixtures [14]. These methods are mostly based on the Hapke model [15]. These models usually need to be tuned to give satisfactory results, and are not always satisfactory [16].

Data driven methods usually are based on constrained dimensionality reduction or classifiers. It is possible for example to do a constrained non-negative matrix factorisation [17].

A important difficulty occurring when using unmixing models is that they require a spectral library containing spectra which are independant of each other, otherwise there are multiple combinations that give similar resulting spectra.

1.2.4 Neural networks

Artificial neural network (ANN) is a form of supervised learning, which has been applied to numerous types of problems [18]. Artificial neural networks are inspired by biology [19], altough a more formal approach is usually more suitable. We quickly summarize such an approach in section 3.2.1, so this part will focus more on the intuition and the general ideas of neural networks.

The perceptron, which is the "base element" of a neural network, was introduced in 1958 [20], and it has been used in artificial neural networks since then. The interest of using neural networks has varied over time for various reasons, but it has risen significantly since the years 2006 [21].

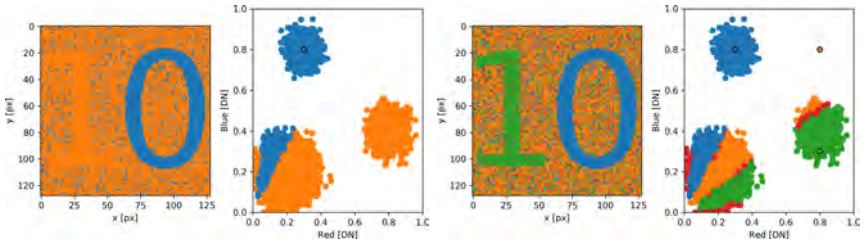


Figure 1.9: Illustration of the spectral angle mapper. The first picture shows only one class. Colors correspond to the class. Red points non classified. The points circled in black are the reference spectra.

Moreover, there is as of 2018 a strong mediatic coverage (although it is often referred as "artificial intelligence").

There are many applications of ANN. We will focus on one common use, when they are performing as predictors. For example, they can be used to predict to which class a spectrum corresponds. They could also be used to perform unmixing of a spectra, in which case the output would be the ratio of each component.

In that configuration, the goal is to find a relation (a parametric function) between input data (the spectrum) and the output data (the prediction of interest). The function structure should be provided, and the goal is to adapt its parameters to get the best possible predictions. To find these parameters, a training set which contains both inputs and the corresponding known outputs (ground truth) is used. This is a type of problem commonly found in science (model fitting), but the specificity of neural network is that the model is very generic, and contains a very large number of parameters. An ANN consists of a multiple operators applied successively, as depicted in figure 1.10. The characteristics of the neural networks are defined by the number and the types of operators used. The size of the intermediates vectors as generated by the operators is usually called "layer size". This size depend, of course, on the operators selected.

A common neural network structure uses affine operators (multiplication by a matrix, and a vector is added) as combination operators, and Rectified Linear Unit (ReLU, replaces any value less then 0 by 0) as activation operators. In this situation, all the coefficients of the matrix and the vector of each of the combination operators are parameters which need to be found by training. Depending on the size of layers, this kind of network has a very large number of parameters.

To do the training, a loss function has to be defined. This function compares the prediction of the network to the ground truth, and returns a value proportional to the difference between them. This means that a

neural network which makes good predictions will have a lower loss value. It is also possible to penalize the result of the loss function if the network's parameters are not as expected. For instance it is possible to use this approach to increase the sparsity of the matrices.

Training a neural network consists in minimizing this loss function for the whole training dataset. There are also many different optimizers which can achieve this, the simplest being gradient descent. Once the loss is low enough (or doesn't decrease any more), the network is considered to be trained, and can be applied to do predictions. Predictions are made by applying the operators with the parameters found during training.

Training an ANN is challenging, as many different hyperparameters (parameters such as the type of operators, the size of the intermediate vectors, the loss function, the type of optimizer etc.) can affect both the efficiency of the training and the accuracy of the resulting neural network. Two common issues are over-fitting (the neural network performs well on the training dataset, but is inaccurate for any other data) and the network learning "dying out" (bad results on the training dataset, and doesn't improve).

To improve the quality of the evaluation of an ANN, the following strategy is generally used:

- The dataset containing ground truth is split into three datasets: the training dataset, the validation dataset, and the test dataset. It is important that those three datasets have the same data distribution.
- The training occurs on the training dataset, but the monitoring of the training process is done by monitoring the accuracy of the neural network on the validation dataset. This limits the risk that overfitting would be unnoticed.
- Once the network is trained, it is tested using the test dataset (which was not used in the training). This helps to limit the chance that the hyperparameters were optimized specially to increase the learning on the validation dataset.

The main advantage of neural networks (and probably what makes them popular nowadays) is that it is possible to get accurate predictions, without having to define a specific model. The quality of prediction can easily be better than the other approaches available.

However, their use is also challenging. First, the choice of hyperparameters of the network is currently an open problem, as there is at present no complete theory about how to select the best parameters for a given problem. There is also large variability in the network predictive capabilities depending on the way the input data is preprocessed. An additional difficulty is that the training requires a large amount of data, with adequate

diversity. If such data is not available, the ANN won't be able to generalize, and therefore won't be able to perform predictions for data which is not very similar to the training data.

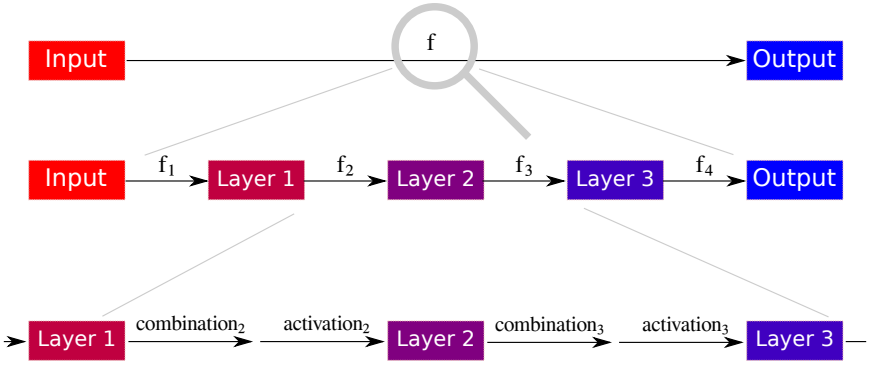


Figure 1.10: Illustration of a neural network seen as a combination of operators. The input, output and data layers are vectors.

1.3 Contribution of this thesis

Mineral identification using remote sensing techniques is challenging for multiple reasons. Therefore, this thesis focuses on the acquisition procedure and the methodology. It consists in multiple articles, which are grouped in this document.

The spectral differences between different samples are small, so it is of key importance to have high quality acquisition procedures. Chapter 2 presents an improved process of laboratory hyperspectral image acquisition. It is a guide on how to capture high quality hyperspectral images in the laboratory, thus ensuring the reproducibility of the data acquisition. It identifies pitfalls in aspects such as the scene setup, illumination, dark frame subtraction, sensor calibration, focus, high dynamic range imaging, and translucent objects. It shows how to evaluate the characteristics of the hardware, how to assess the influence of the choices made during the image acquisition and provides strategies to improve the quality at each step, without requiring additional hardware.

Unmixing is difficult, as the spectra in libraries are not independent enough. The only possible approach to avoid very high uncertainty in the unmixing result would be to use a subset of the spectral library, but the choice of the particular spectra used as reference spectra is not straightforward, as a mineral can have different spectra depending on the sample and its preparation. Therefore, an approach based on classification using

artificial neural networks was selected. However, artificial neural networks are not usually designed to perform on spectral data with missing bands, which is a situation that often occurs due to saturation or bad pixels. The second article, in chapter 3, presents a novel approach about make neural networks more robust when handling data with missing values.

A large amount of training data is required to use a neural network classifier, which was not available. The third article, in chapter 4, describes the process of building such a hyperspectral library, which can be used to train a neural network classifier.

Additionally, a lot of work has been done which is not directly in article form. A significant portion of the work done for this thesis was to create a set of efficient tools to acquire and process hyperspectral data. The details about this process and design choices are shown in chapter 5. A short user manual is also provided in appendix A. Work has also been done on the determination of surface water content of bentonite. An application of this technique has been done as a technical report for NAGRA, which is in appendix B.

Chapter 2

Improving laboratory short-wave infrared imaging

This paper proposes a guide for capturing high quality hyperspectral images in the laboratory and ensuring the reproducibility of data acquisition. The paper indentifies several pitfalls during image acquisition such as the scene setup, illumination, dark frame subtraction, sensor calibration and focus. It also provides a method to perform high dynamic range imaging, and to acquire data on translucent objects. It illustrates and explains how to evaluate the characteristics of the hardware, how to assess the influence of the choices made during the image acquisition and provides strategies for improving the quality at each step, without requiring additional hardware.

Introduction

Hyperspectral imaging has been widely used in recent years. While (point-wise) spectrometers were invented in the 17th century [22], it was only in the second half of the 20th century that imaging spectrometry was developed [23]. This technique enables measuring simultaneously spatial and spectral information, eventually yielding an image with spectral information for each spatial point. Spectral information is represented by bands, which are described by their center wavelengths, and their "width". Imaging spectrometry became well-known in 1972 with the satellite ERTS-1 (renamed Landsat-1): hundreds of papers were published using the data (for example [24], [3], etc.).

Over the years, both spatial and spectral resolution improved, going

from a few "discrete" bands to hundreds, which is considered an approximation of a continuous spectrum. To the best of our knowledge, the term "hyperspectral imaging" was first used by Goetz in the eighties [25], and is becoming quite common recently (fig. 2.1).

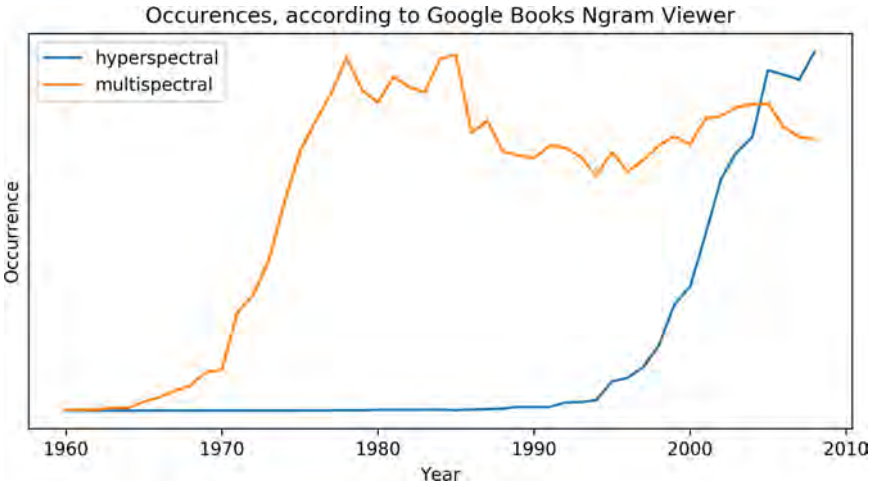


Figure 2.1: Comparison of the occurrence of the terms "hyperspectral" and "multispectral" in the Google Book corpus, per year of book publishing date.

In the beginning of the 21st century, hyperspectral cameras have become more affordable, making them accessible to more and more research groups, and thus allowing scientists to not only process existing data, but also to acquire their own data. Benefiting from computing power availability, numerous algorithms to process and classify data have been developed, such as PCA [6], MNF [5], SAM [10], SMACC [26], PPI [27], Voronoi classifiers [28], etc. These factors have led hyperspectral imaging to be widely used in laboratory setups, for example for mineral identification [29, 30], the food industry [31, 32], the health industry [33], archeology, etc.

These papers focus on the analysis of the acquired data, and few details are given on the acquisition procedure. However, acquisition should be carefully thought out due to the technique's complexity. As Geladi et al. [34] notes, "Bad sensors, nonlinearity of sensors, differences between sensors, inhomogeneous illumination and limited resolution of the A/D converter all influence the results". Several papers also identify some difficulties, like Moscetti et al. [32], who states: "Positioning of the hazelnut classes on the sample holder was randomly permuted at each scan (120 iterations) to avoid misleading statistical inferences due to camera artifacts and pixel misregistration problems."

The similarity of the procedures described and the fact that most papers don't cite any source for their procedure lead us to consider that the procedures generally originate from the camera manufacturers' user manual, or from general knowledge. The procedures generally consist of 6 steps:

1. Record dark frame D
2. Illuminate the scanning area evenly
3. Focus, and fix integration time
4. Record a white frame W
5. Scan data frames F
6. Compute reflectance as $r = \frac{F-D}{W-D}$

Usually, these steps are executed using software provided by the manufacturer. There are small differences between the procedures proposed, mostly in the third step. For example, focus is done either by using a mire (black-white stripes) or directly on the subject, to have the image visually sharp. For the integration time, some use automatic algorithms, while for others it is up to the operator to choose a value which avoids saturation, while keeping an acceptable signal-to-noise ratio. Researchers also make various choices for illumination, focus, integration time, subject disposition, etc. However these choices influence the results' quality and reproducibility, it is therefore important and challenging to understand the implications of theses choices and select proper procedures.

The goal of this paper is to provide for each aspect of the image acquisition a theoretical background, an identification of pitfalls, and a procedure to improve the quality.

Each section addresses one of these aspects, and provides comprehensive advice to improve the quality of the data acquired: imaging setup (section 2.1), dark frame subtraction (section 2.2), non-uniformity correction (section 2.3), focus (section 2.4), high dynamic range (section 2.5) and finally details on how to capture data from translucent objects (section 2.6).

Each section consists of three parts: the first part gives a general introduction to the problem, the second part the proposed approach, and finally an application example acquired using a Specim shortwave infrared (SWIR) hyperspectral camera.

System overview and setup

The camera used here is a Specim shortwave infrared, it is a line scan camera of 320 spatial pixels. For each of them, the camera captures 256 wavelengths between 0.9 µm and 2.5 µm. However, the methods presented in this paper are applicable to all hyperspectral imaging devices.

Camera sensor output (the measured "value" of each pixel) is usually expressed in counts, or "digital numbers" (DN). We will additionally use the following non-standard units, to be more explicit:

- DN_r: Raw digital numbers from the sensor
- DN_d: Digital numbers of frames on which dark frame subtraction has already been applied.

Throughout this document, upper case letters will be used for measured values, while lower case letters represent "true" values. For example, W is the white frame captured by the device, whereas w is the white frame that would have been captured if the camera was flawless.

2.1 Illumination and imaging setup

Despite its similarity with photography, setting up hyperspectral imaging has its own set of challenges, mostly related to illumination. The ideal illumination would be of high intensity and homogeneous, both in intensity and spectrally, and cover the spectral range of the camera. Even then, illumination is a complex topic, as light can follow multiple optical paths between the source, the sample, and the sensor. Modelling these paths is usually challenging, especially since the characteristics of the sample are unknown. However, the effects encountered can be approximated as a combination of (fig. 2.2):

- Specular reflection, which is the bright mirror-like reflection that can be observed on objects. It usually occurs on the points of the sample which are oriented in such a way that the object behaves like a mirror. Specular reflections are spectrally usually very similar to the incoming light.
- Backscattering, which is the general light coming in return of the object, with the exception of the specular reflections. In visible light, this corresponds to the "color" of the object.
- Transmission, which is the light going through the object, for a translucent object. In visible light, this is perceived as the color of the object when it is situated between the light source and the observer.

Diffusion is the light resulting from both specular reflection and backscattering. Additionally, the term reflectance is used to express the physical characteristics corresponding to the ratio between light diffused back and incident light on the sample. In the context of hyperspectral imaging, this usually excludes the specular reflection, and therefore consists in the ratio between the backscattered and the incident light. A more precise description, which takes into account the position of the light source and the position of the observer, is given by the bidirectional reflectance distribution function

(BRDF). It has the advantage of also describing specular reflection, but is much more complex [35].

Similarly to reflectance, transmittance corresponds to the ratio of light which is transmitted by an object. When working in a hyperspectral context, reflectance and transmittance values are measured per-wavelength, resulting in a transmittance spectrum or a reflectance spectrum, instead of scalar values. The characteristic of interest is usually the reflectance spectrum (restricted to the backscattering), therefore the challenge is to separate the various optical effects. Section 2.6 proposes a method to scan translucent objects.

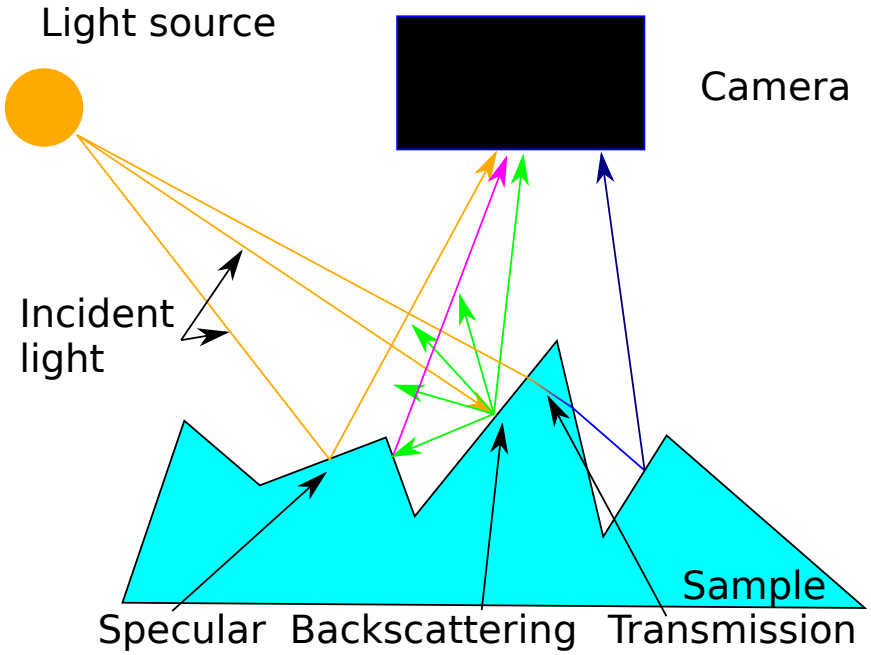


Figure 2.2: Ilustration of the various possible optical paths.

2.1.1 Decreasing illumination artifacts

To measure the reflectance, it is required to measure, for each point, both the light backscattered by the sample and the incident light.

Measuring light reflected by the sample is usually straightforward, apart from specular reflections. These reflections may decrease significantly the signal-to-noise ratio in the areas where they occur. The simplest way to reduce occurrence of specular reflection is to arrange the light source

position, the sample and the camera in such a way as to minimize the areas in which the surface of the sample behave like a mirror.

Multiple light sources are commonly used in published papers (for example, [36] use 18 halogen lamps), as they decrease the number of shades and visually improve the homogeneity of the illumination. However, multiplying the number of light sources increases the number of optical paths, and increases the total area of specular reflections. Moreover, spectral differences between lights (due for example to differences in the filament or in the voltage provided) may lead to errors in measurement. Therefore, it is generally better to only have one light source, as punctual as possible, especially since shades are easier to identify than specular reflections.

Another issue with samples which exhibit high reflectance or high transmittance is that the light can be diffused twice consecutively, or can be first transmitted then diffused, which increases measurement errors. Unfortunately, there is no simple way to avoid these kinds of issues, but modifying the positions of the objects can reduce the likelihood of such optical paths.

It is in general not possible to precisely measure the incident light for all points at the sample position. As a substitute, an object of known reflectance can be positioned near the sample. By dividing the measured light from this object by the reflectance of the object, it is possible to get an estimate of the incoming light, which is considered to be valid for all points of the image. Most researchers use a white reference, like a Spectralon, which is considered to be a perfect diffuser in all wavelengths.

Unfortunately, this setup only yields an approximation of the real reflectance. To ensure its quality, it is required that:

1. Only one light source is used.
2. No diffusion occurs on other objects on the scene, in particular on bright objects, as this would behave like another light source of different spectrum.
3. The light source emits light with the same spectrum on all points of the mesurement area (spatial light uniformity)
4. The light source emits light which is stable over time, both in spectrum and in intensity (temporal light stability)

Additionally, to maximize the signal-to-noise ratio, it is advised to use a light source with maximal intensity at the wavelengths of interest.

A carefully thought out setup can limit the influence of light diffusing on other objects. We will show examples in the next section.

For the two last points, it is necessary to evaluate the spatial and temporal characteristics of the light. This requires an object to visualize the light flow, either in transmittance (object is between the light source and the camera) or in reflectance (object is illuminated by the light source as

usual). This object should be uniform in order to know that any difference is due to illumination.

A teflon plate can be used to evaluate the spatial uniformity of the spectrum, as the plate is uniform, and is translucent. In the example shown in figure 2.3, the transmittance of teflon plate is used to measure the spatial uniformity of the light source. Once the characteristics of the light are known, it is possible to assess whether the sample is correctly illuminated or not.

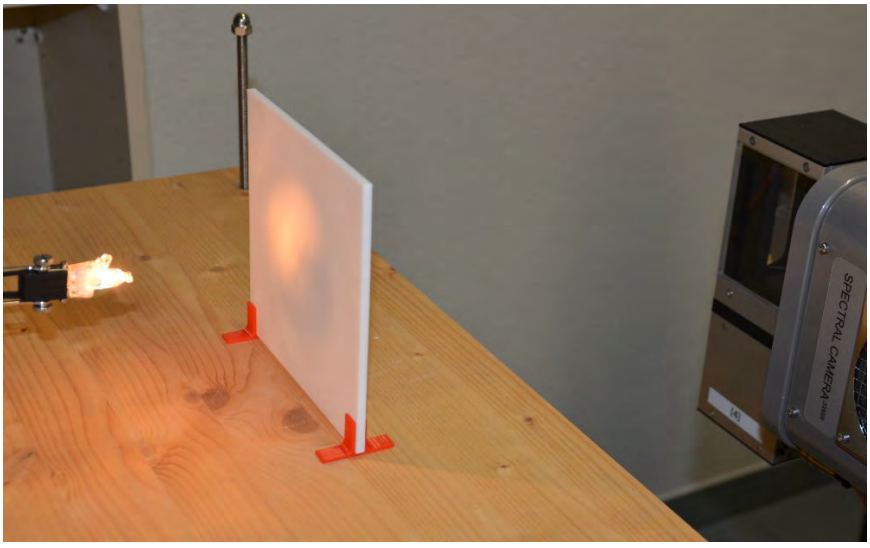


Figure 2.3: Setup to measure the stability of the illumination. A teflon plate of 4mm width is used as target. Since it is translucent and homogeneous, measuring the transmitted light gives directly information on the light emitted by the source.

After an initial warm-up time, the stability over short time spans of a light source depends on the power source, and on the interaction of the provided current frequency and the camera frame rate.

Ideally, using direct current (DC) is better, since DC is inherently temporally "stable", but high power DC sources are not always available. Therefore there is a choice to be made: either use high power lamps, but with alternative current (AC), or use low power DC bulbs, which are more "stable" but provide less light.

Therefore, it is interesting to evaluate how "unstable" the AC lights are. This mainly depends on the thermal inertia of the light source. The bulb still emits light for a while even if there is no effective current going through the filament, because the filament requires some time to cool down.

Of course, if the frames of the camera were captured exactly at the same moment of the cycle, the conditions would be exactly the same. This could be achieved by setting the frame rate to be a divisor of the frequency of the electrical power network. However, since the camera has its own clock for the frame capture, and is therefore not synchronized with the AC power sine (which frequency also varies over time), some shifting in the cycle may occur. A theoretical example can be seen in figure 2.4.

A variation of intensity is not usually a problem, since most of the time only the reflectance shape is of interest (and therefore each pixel can be normalized by its intensity), but a variation of spectrum would create issues. To measure the variations, a setup like the one in figure 2.3 can be used, with a framerate slightly different from the frequency of the AC, to maximize the variability.

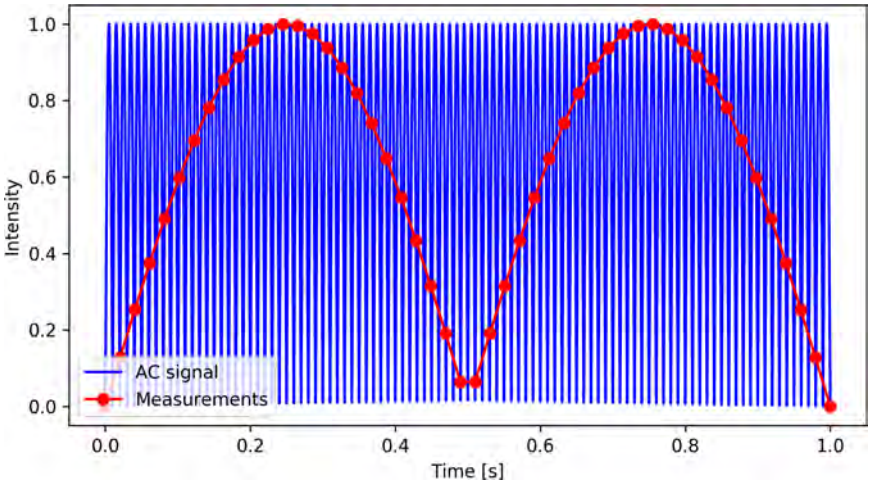


Figure 2.4: Theoretical light intensity which would be observed by measuring a light source without thermal inertia, powered by an AC signal of 50Hz, by a camera capturing 49 frames per seconds. It results in capturing each part of the cycle every second. Since only the voltage matters, not its direction, two period per seconds are obtained.

2.1.2 Application example: evaluating light source characteristics and impact of the scene setup on the measurement

The first part of this section will be about practical evaluation of light sources, while the second part shows how diffusion on surrounding objects can affect the measurements.

A suitable light source should provide a continuous spectrum, which incandescent light sources usually have. While specialized SWIR light sources exist, off-the-shelf halogen lamps are widely used, due to the fact that they are inexpensive and provide a continuous spectrum. Therefore they will be investigated throughout this section. Although they do not seem widely used, quartz infrared heaters will also be investigated, as they have a higher output at longer wavelengths, therefore are complementary to halogen lamps.

Halogen lamps can be powered by either alternative current or direct current.

The light sources investigated are summarized in figure 2.5, the setup used is presented in figure 2.3.

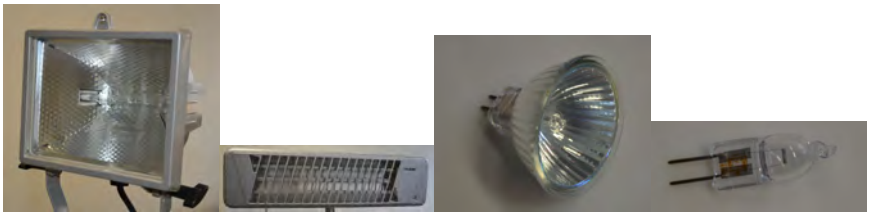


Figure 2.5: Light sources investigated. From left to right: 230V AC 300W halogen flood light, 230V AC quartz heater, 12V DC 30W halogen lamp with reflector, 12V DC 30W halogen bulb.

2.1.2.1 Light source evaluation

First, it should be noted that the requirements depend on the subject, and the criteria of how stable and how homogeneous a light source has to be depends on the sample. For example, if the spectral variability of the subject is quite significant, a small inhomogeneity of the light can be tolerated, while when looking at sample with low variability, high quality illumination is required.

First, let's evaluate how "unstable" the lights sources are. As expected, the DC lights are very stable.

Since the AC light sources are powered by a 50Hz electrical network, the camera was set to capture 49 frames per second. The results for a single pixel are shown in figure 2.6 and 2.7.

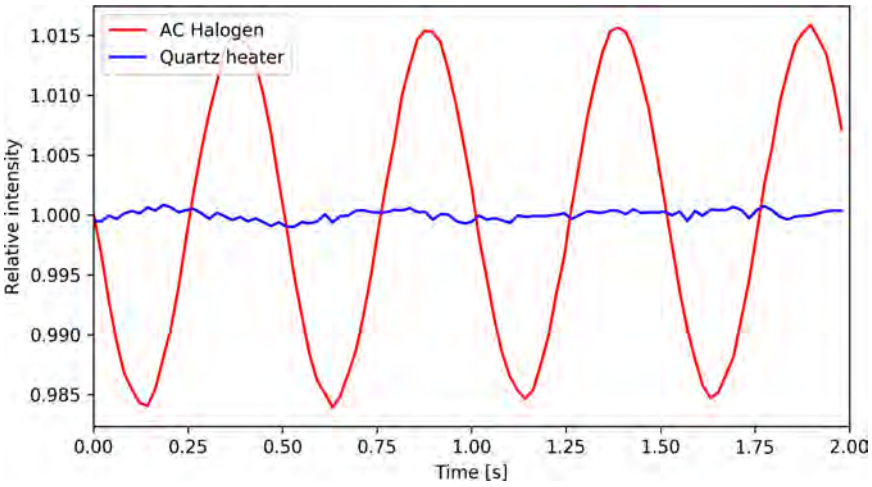


Figure 2.6: Intensity of the light as a function of time. The intensity is normalized in order to have its average over time to be 1. Note that the shape of the halogen is as predicted, while the quartz heater does not seem to be influenced directly by the AC frequency. This is due to the large inertia of the heating element.

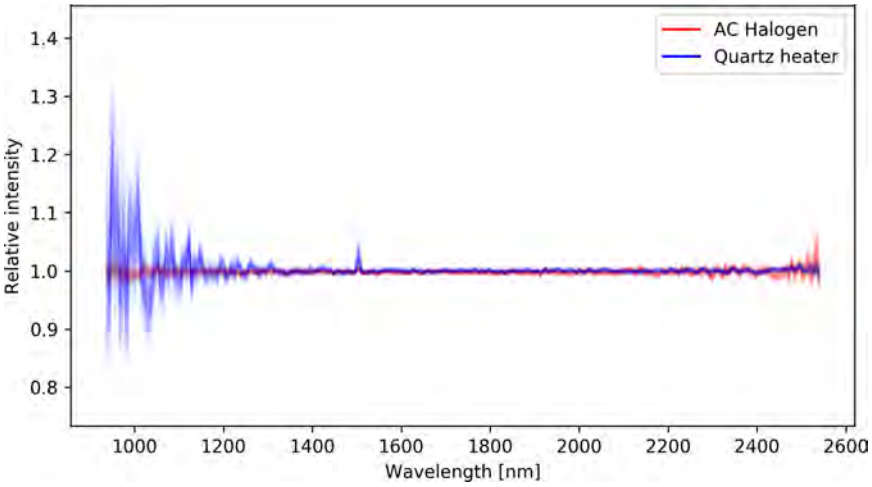


Figure 2.7: Evolution of the spectra over time, where each spectrum is divided by the average spectrum over time, in order to show its relative variation. One can see that there are slight variations of the spectrum, but this is due to the noise of the sensor, and not to the light sources.

In conclusion, the AC halogen lamps can be used as long as the intensity is not a factor, while this AC quartz heater can be used without any issue, as the thermal inertia of the heating element completely smoothes the variations due to AC.

Spectral homogeneity cannot be assessed by the naked eye (except in extreme cases), however, using the setup previously described (fig. 2.3), it is simple to measure. In general, for large lamps, the lamp is far from the sample. It would be tempting to use lamps with reflector to focus the light on the sample, but we can see in figure 2.8 that the reflector creates big spatial inhomogeneities in the light spectrum. The orientation of the light bulb also has a large impact.

The following general observations can be made:

- Reflectors can cause problems (fig. 2.8)
- For a simple light bulb, the best results are obtained when orienting it in such a way that the filament is as in figure 2.5. Indeed, the light is emitted by the surface of the filament, which is greater from the front and more homogeneous than on its side. Moreover, the top glass "blob" disturbs the light.
- Since the emitted spectrum depends on the temperature of its source, the emitted spectrum is not the same in the different parts of the bulb. It is therefore important to orient all the bulbs in the same way with respect to the sample.
- For halogen flood lights, the spectrum is mostly homogeneous in the center of the light beam. Given that these kind of lamps usually have a high power, it is generally sufficient to have the light far enough to have an homogenous light.
- The spectral homogeneity of the quartz heater considered is good, but it takes a few minutes to stabilize. One of its drawbacks is that it does not emit visible light, making it more challenging to orient.

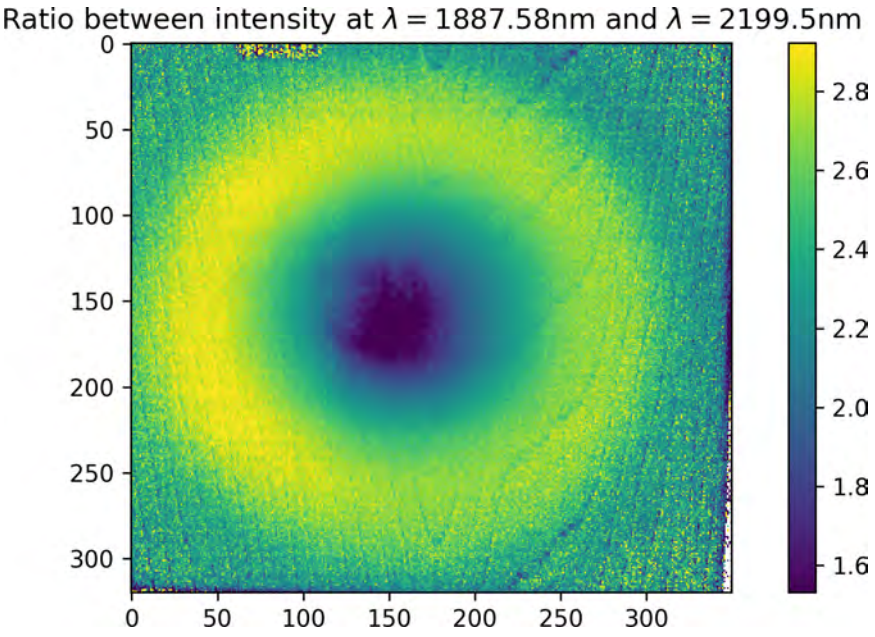


Figure 2.8: Wavelength ratio for the halogen lamp with reflector. Although the light spot generated by the lamp seems homogeneous when observed, a circular pattern is clearly visible in infrared.

2.1.2.2 Impact of diffusion on the measurements

In a measurement setup, the sample is illuminated by both the light source and all the light diffusing from any object near the sample. Diffusing on surrounding objects affects the measurements on the sample, as the diffused light has the "color" of the object it diffused from. In this section we will investigate the significance of this effect.

Two different setups are commonly used when acquiring data. Either the camera is standing in front of the sample (fig. 2.9) or it is above the sample (fig. 2.10)

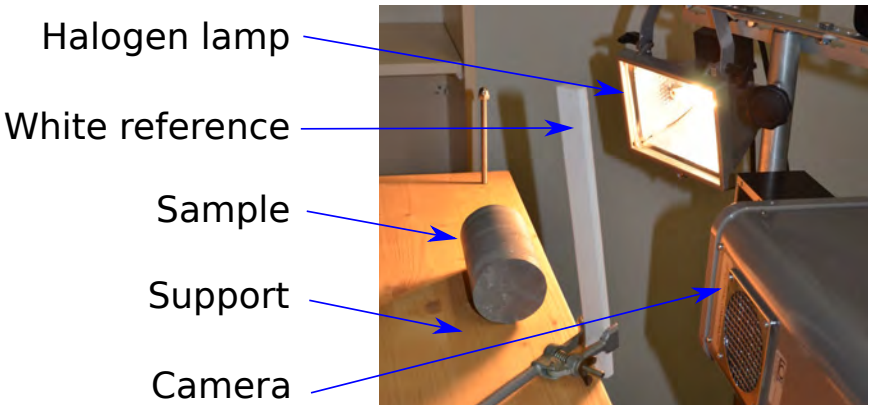


Figure 2.9: Typical setup when imaging from the front.

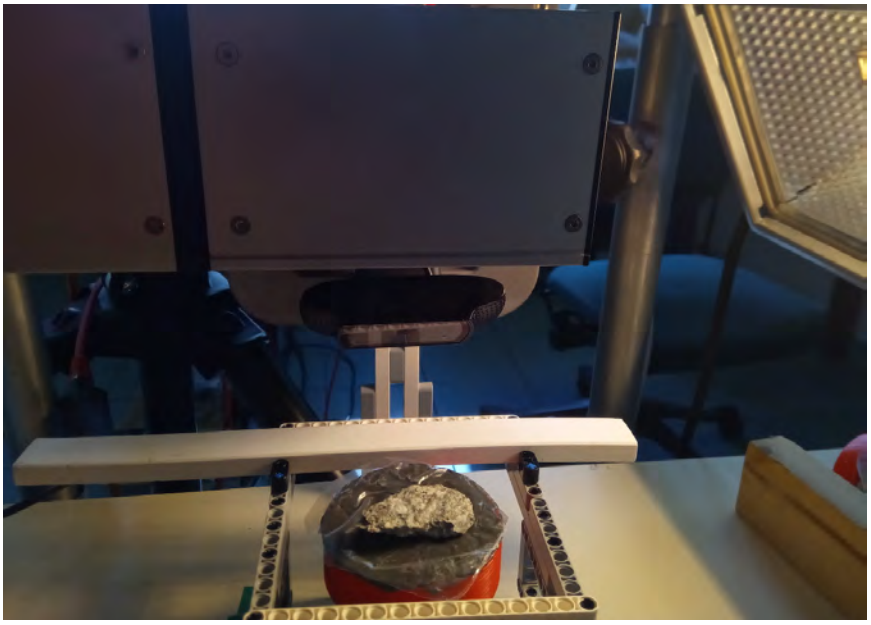


Figure 2.10: Example setup when imaging from above. The sample is put in a holder containing black modelling clay. There is a plastic sheet between the modelling clay and the sample, to avoid contaminating the sample. This is not the typical setup in the sense that there is also a webcam which is used for visible light photogrammetry.

Imaging from the front is straight forward (fig. 2.9). It has the advantage

that there are no difficulties changing the positions of the sample, the camera, and the light sources. However, the support on which the sample is placed may influence the measurement. The same applies to any object which is in the surrounding of the sample.

As an example, let's consider the setup shown before. Figure 2.11 shows the setup from the camera view point, and figure 2.12 shows the image in infrared.

Figure 2.11: Sample as seen from the camera. The core has a diameter of 7cm, and is 12cm long. The support is made of Epicea wood.

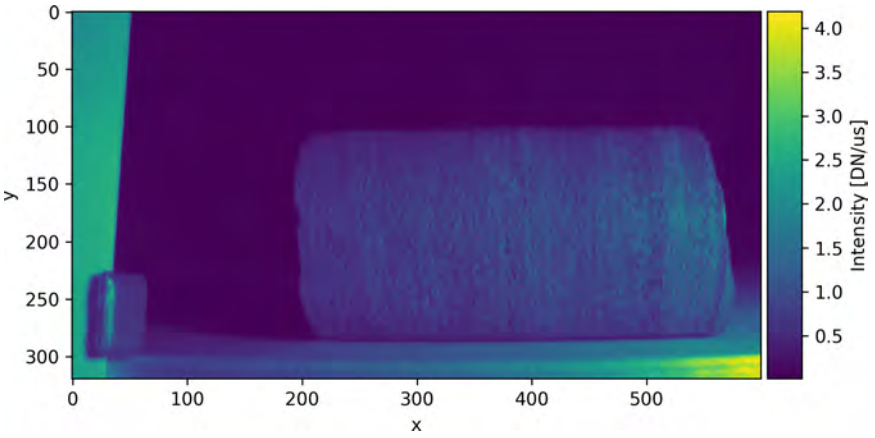


Figure 2.12: Average intensity of the shortwave infrared bands.

Both in visible light and in SWIR, there are no obvious artifacts. However, if we pose a black sheet horizontally on the wood in front of the sample, the measured spectra change significantly (fig. 2.13), as the diffusion of the black sheet is significantly smaller. The black sheet was choosen because its reflectance is very different from the one of the wood, both in shape and in magnitude.

To see the extent of such change, we can compute the difference in intensity between the image with wood background and the one with the black sheet of paper. The results are shown in figure 2.14. As we can see, this affects large areas of the scene.

It important to note that every illuminated object on the scene backscatters some light, and that usually this light is not of the same spectrum as the light source. For example, even a white Spectralon reference does not have a perfect reflectance, it reflects less light at 2100nm for instance than at 1500nm. This may result in errors in measurement due to the presence of the white reference.

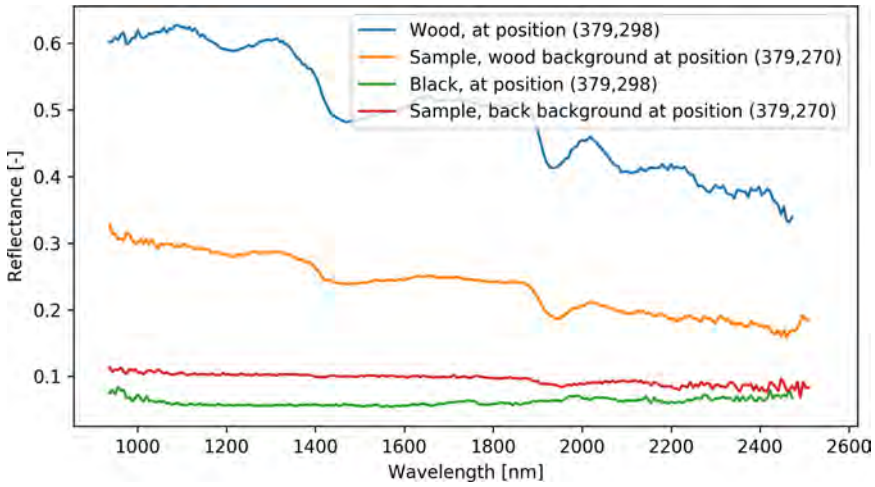


Figure 2.13: Comparison of the spectra measured on the sample with two different supports. As we can see, the sample point is significantly influenced by the spectrum of the neighboring background point, due to diffusion.

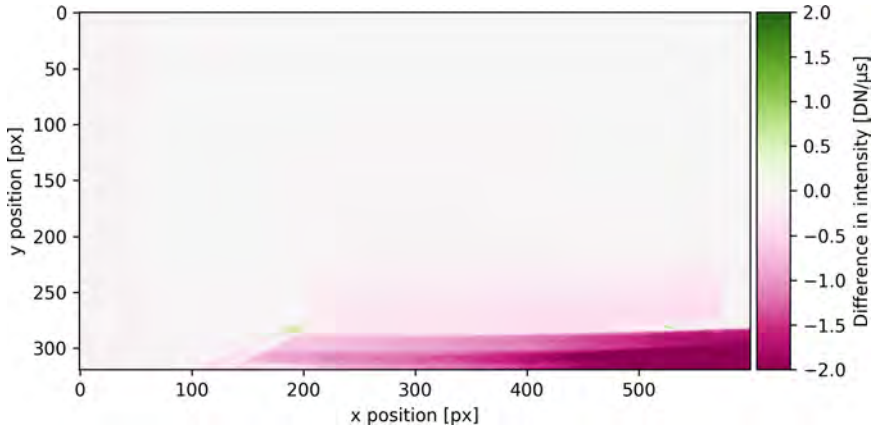


Figure 2.14: Difference of SWIR intensity between the wood and the black sheet. The difference has been computed on HDR images, therefore the units are in DN μs^{-1} (see section 2.5). We can see that the diffusion not only affects the sample, which is near the sheet, but also the corner on the white reference.

Unfortunately, there is no perfect and simple solution for this issue. One of the solutions is to isolate the sample from any other object in the

surrounding, for example by hanging the object instead of putting it on a support. This gives good results since the light intensity is decreasing proportionnaly to the square of the distance, but may cause practical problems, depending on the sample.

It is also possible to paint every object in the scene black, however the various black pigments usually still reflect a few percent of the incoming light. Moreover, black objects tend to heat up quickly, which may create issues if the sample is heat sensitive, or even create fire hazards.

On the other hand, doing hyperspectral imaging from above has multiple advantages. It is possible to scan objects that are not able to stand, like powders. Moreover the surrounding of the object is easier to define (e.g. put the sample on a black sheet of paper). However, in addition to issues with the surrounding, there are also some constraints related to the positioning of the light.

Since the camera is usually close to the sample when imaging from above, the light has to be on the side with quite an angle, which creates more risk of light transmission, if the sample is translucent.

2.2 Dark frame subtraction

A CCD sensor can be understood as an array of photo sensitive detectors, coupled with electronics to obtain a digital value corresponding to how much light reached each of these detectors. This electronics is an analog-to-digital converter (ADC). An perfect sensor would return values proportional to the number of photons reaching each pixel, however in practice various sources of noise affect the results. Moreover, the ADC can have an offset, in the sense that all the output values will be shifted.

The main source of noise is the dark current, which adds a different offset to each pixel, as shown in figure 2.15. This current is due to electrons which “seep” into the detector due mostly to thermal effects, and increases the value of a pixel, even in the absence of light. This noise can be seen as a pattern in the image, and is called the dark signal fixed pattern noise. It is mostly independant of the amount of light received by the sensor, and is added to the signal due to the light. In some conditions (low illumination or high integration time), this fixed pattern noise can be greater than the signal due to the light. Most imaging systems usually subtract this noise, which implies that is important to estimate its value correctly.

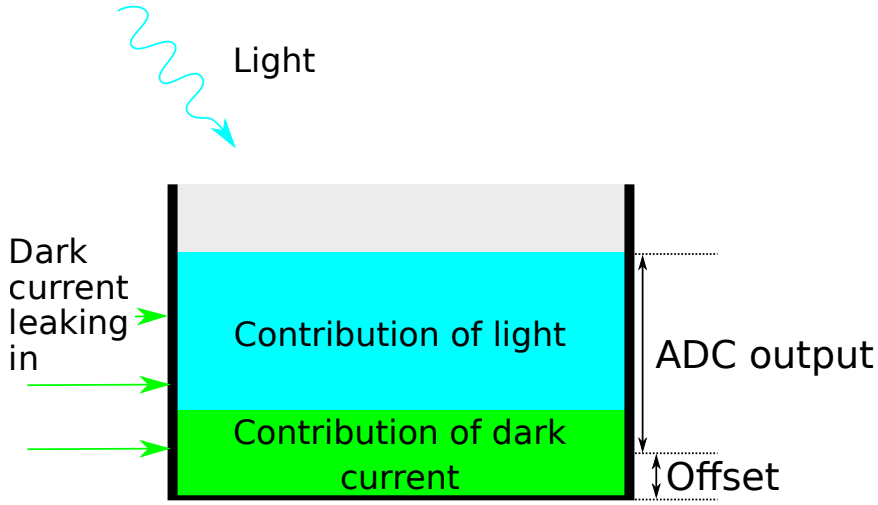


Figure 2.15: Illustration on how the dark current and the offset influences the read out value of the ADC. The measurement is expected to be only the contribution of light, but dark current increases this value. Additionnaly, the ADC may also offset the value. Usually, dark frame subtraction cancels both the contribution of the dark current and the offset, to measure only the contribution of the incoming light.

Shortwave infrared hyperspectral cameras are usually more affected than visible light camera, due to the fact that the incoming light has to be divided into spectral bands.

In principle, this noise is easy to estimate since it is what remains when no signal is measured, by taking a picture with the shutter closed (therefore the name “dark frame”), and then to substract this frame to any data frame capture afterwards. This is a very commonly used technique, named “dark frame subtraction”.

In practice, two technical difficulties may arise.

First, dark frame subtraction assumes that the dark frame subtracted matches exactly the dark current, otherwise it would induce a systematic error. As the temperature of the sensor evolves over time, it would be optimal to capture the dark frame simultaneously with the data frame. Since the pixel cannot measure simultaneouly its value with and without incoming light, a strategy which yields a sensible approximation should be devised. Such a procedure is provided in section 2.2.1.

It may also happen that, for some reason, good quality dark frames are missing. This can be the case if the camera does not have a computer-controlled shutter, or if errors were made during the capture. Section 2.2.3 provides a method to recover missing dark frame data.

More information about dark current and related techniques can be found for example in [37], [38], [39], and [40].

2.2.1 Factors influencing the dark current

Various factors influence the dark current, the most significant one being temperature. Hyperspectral SWIR cameras usually use active cooling (Peltier element), controlled by some hardware algorithm. It is therefore required to understand how the camera and this algorithm behave. While very advanced techniques exist, for a simple approach the main questions are:

1. How much time is required at camera startup to have stable conditions?
2. How stable is the dark current in the usual experimental conditions? Does it evolve in cycles (hysteresis)?
3. How sensitive is the camera to outside conditions changes?

This information is usually not provided in the user manual, therefore an experimental approach should be taken. A decision should also be made as to how precise the measurements should be, as obtaining a high-precision dark frame requires a more constrained work-flow.

For the first question, a single parameter must be found: t_{startup}. It consists in the time required to get the camera to a steady state. Experimentally, this can be done by plotting the average of the pixel values of the dark frame as a function of the time, smoothing it if needed, and then choosing a "safe" value. The behavior to adopt is very straightforward: one should simply wait more than t_{startup} after the camera was started before making measurements.

For the second question, measurement of dark frames over approximately half an hour should be made. One can expect cycles of length t_{cycle}, which depend on the cooling algorithm. There are various strategies to estimate the dark frames, depending on the constraints of the experiment being performed. For short experiments (less than a third of t_{cycle}), it is a valid approximation to capture capture a dark frame before and after the scan and to interpolate. For longer scans, the simplest approach is to consider the variation of the dark frame as noise without any structure.

The last question requires making long term measurements of the dark frames on the camera while influencing the environmental conditions (increasing temperature, adding ventilation, etc.). The impact of these variations cannot be fixed by software, but it can provide useful insight about how the experimental setup should be designed. For example, it can make sense to move the illumination source further away from the camera, or to add a fan to circulate the air around the camera.

2.2.2 A strategy to minimize error in dark frame subtraction

It is important to verify that the main factor influencing noise is temperature. This can be done by measuring the power consumption of the camera and comparing it to the noise level. The results for the Specim camera are shown in figure 2.16. As we can see, there is a clear relationship, indicating that the noise level evolution is indeed related to cooling. There are also noise variations in the fan that corroborate this finding. We can also see that the dark frame values evolves in cycles of about $t_{\text{cycle}} \approx 55\,\text{s}$.

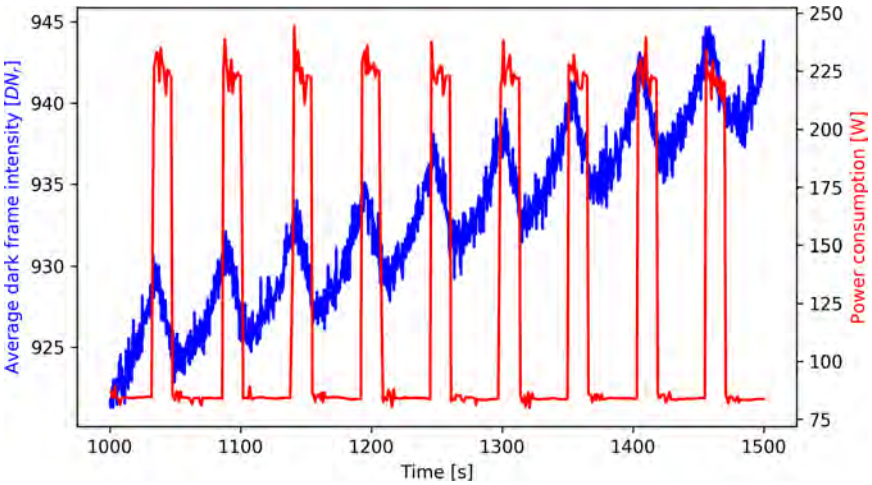


Figure 2.16: Average pixel value of the dark frame compared to camera power consumption. The noise level decreases each time the cooler runs. Then when the cooler stops, after some inertia, the noise level increases again.

First, the startup time of the camera t_{startup} should be measured. We simply capture dark frames regularly from the startup of the camera until it stabilizes. For the Specim camera, this time is about 10 minutes (see figure 2.17). Note that there can still be long term effects affecting the dark frames, so it is not a steady state. This only means that the initial variations due to startup are not occuring.

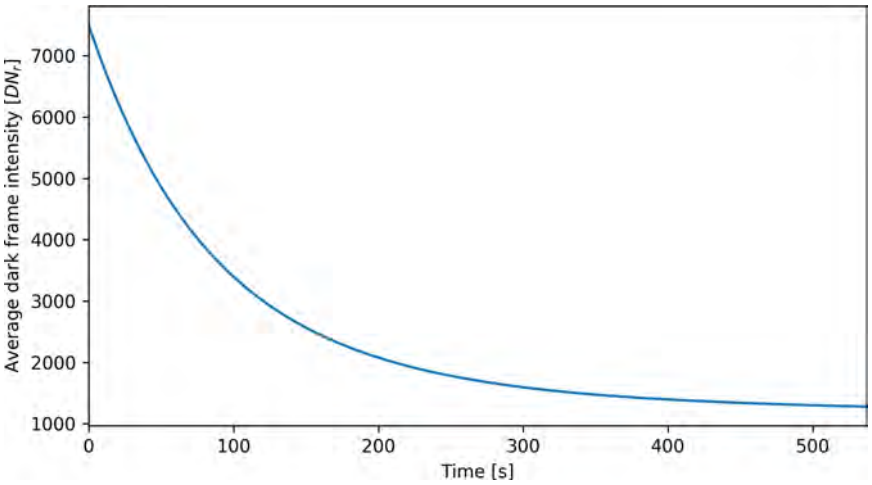


Figure 2.17: Average pixel value of the dark frame at camera startup. The first acquirable frame was shot about 20s after the power was supplied to the camera. It looks smooth due to scale (there are very high values at the beginning). A criteria should be defined as to what is "stable enough", but in practice a manual estimate is often sufficient. For example here it looks stable at the end of the plot, so $t_{\text{startup}} = 600\,\text{s}$ provides a safe margin.

To evaluate how sensitive the camera is to external thermal conditions, experiments were made using a fan and a halogen lamp. For example, the lamp was oriented directly on the body of the camera, effectivily heating it. Then the fan was turned on to provide fresh air. The results can be seen in figure 2.18.

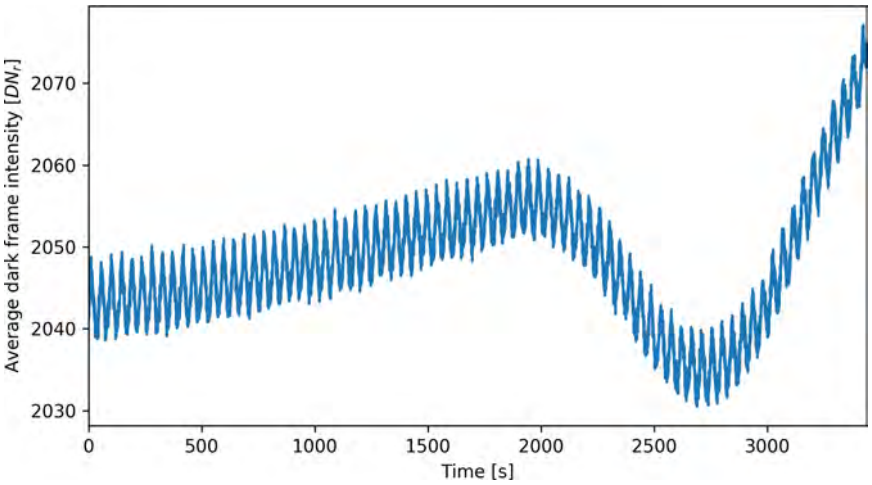


Figure 2.18: Average pixel value of the dark frame when the camera body is heated by a 300W halogen lamp. At $t = 2000\,\mathrm{s}$, a fan was turned on to blow fresh air on the camera until $t = 2600\,\mathrm{s}$. Note that there are both high frequency effects (the Peltier element cooling the sensor is making cycles) and a low frequency change due to the overall temperature modifications.

Therefore, for this camera, the following information can be obtained:

- $t_{\mathrm{startup}} = 600\,\mathrm{s}$
- $t_{\mathrm{cycle}} = 55\,\mathrm{s}$
- The camera is sensitive to variations of the external temperature and there seems to be no monitoring of the sensor temperature.

We can also devise a strategy:

- The camera must be powered on for at least 600 s before performing measurements.
- For scans of a few seconds (a lot shorter than t_{cycle}), capturing a dark frame just before the data frames is a good approximation.
- For scans of about 10 seconds (shorter than $\approx \frac{t_{\mathrm{cycle}}}{4}$), a linear interpolation between the dark frame captured before and the one capture after the data is a good approximation.
- For longer scans, a full cycle of dark frames should be captured before and after the scan, and interpolation of the average before and after is used. Additionaly, the estimated noise added by the incorrect dark frame subtraction can be devised.

The strategies to capture are illustrated in figure 2.19.

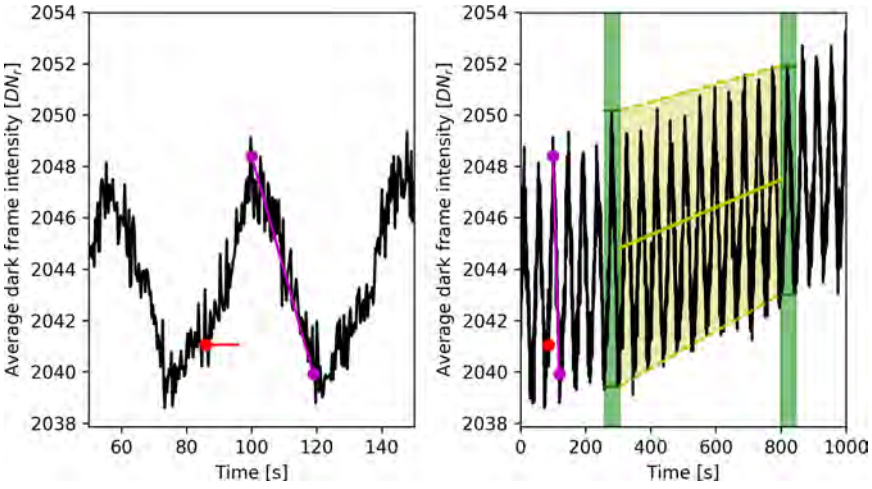


Figure 2.19: Various strategies for dark frame estimation. For strategy 1, in red, a dark frame is captured just before the data acquisition, and is used for each of the data frames captured afterwards. The estimation can be bad if the dark frame changes a lot. In strategy 2, in purple, a dark frame is captured before and after the data frames. It works well in this example but may not work well if the dark noise level is highly non-linear (for example if the time span is too long). Finally, for strategy 3, a set of dark frames are captured before and after the scan (green areas), then the mean value and the standard deviation are computed. Then an interpolation is computed together with a confidence interval, and the interpolation is done (in yellow).

2.2.3 Recovering missing dark frame information

It may happen that dark frames are not captured correctly with the measurements and therefore should be recovered afterwards. Pixels in the dark frame are strongly correlated, therefore they can be expressed as a function of a small dimensionality value. The goal is to recover such a relation and then to find the parameter which yields a dark frame as similar as possible to the one that would have been captured.

For a given integration time, the main factor influencing the dark frame is temperature, as seen in section 2.2. As a consequence, the value of a given pixel of the dark frame can be approximated as a function of the temperature. Temperature is not always available, but since all pixel values are increasing as a function of the temperature, the values of the pixel can

be expressed as a function of the dark frame average m. Figure 2.20 shows values of specific pixels as a function of the average.

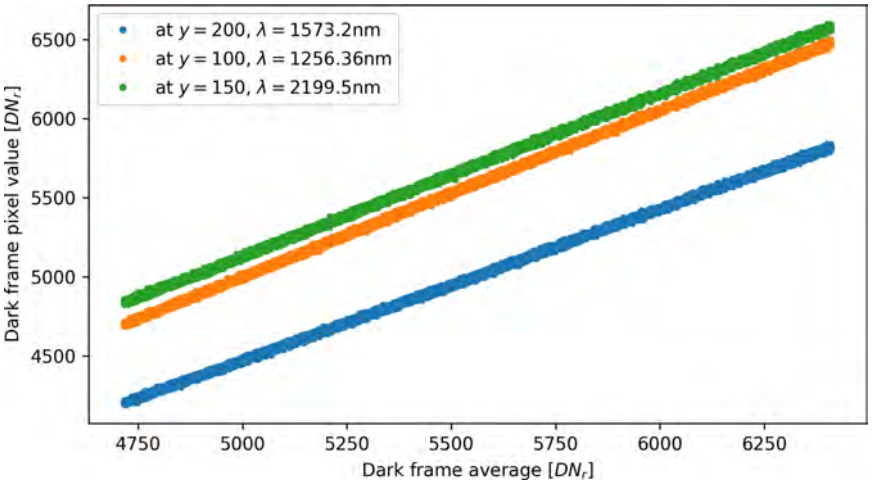


Figure 2.20: Dark frames values of three specific pixels as a function of the average value of the dark frame. There is clearly an affine relationship. Note that both the intercept and the slope vary depending on the pixel.

Given the linearity of this relation, a linear regression is appropriate to find the coefficients α and β:

$$d_{y,\lambda}(m) = \alpha_{y,\lambda} \cdot m + \beta_{y,\lambda} \tag{2.1}$$

The results are depicted in figure 2.21.

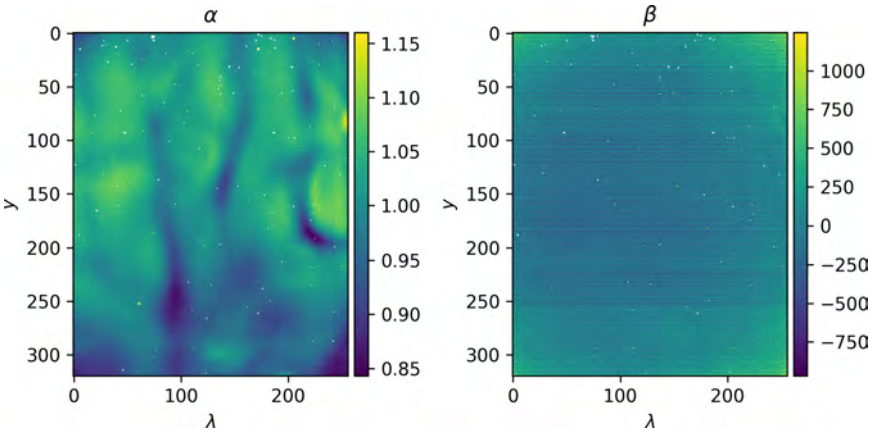


Figure 2.21: Coefficients α and β found by linear regression, for an integration time of 15 000 µs. y dimension corresponds to the spatial pixels, while λ dimension to the wavelengths.

Since the captured data is the sum of the "real" signal, and the dark frame, it remains to find the temperature m that gives the best results. Multiple approaches are possible. If we assume that the dark frame is constant during a scan, one approach is simply to minimize the frame spatial variance, as the inter-frame variance is not affected by adding a constant offset to the values.

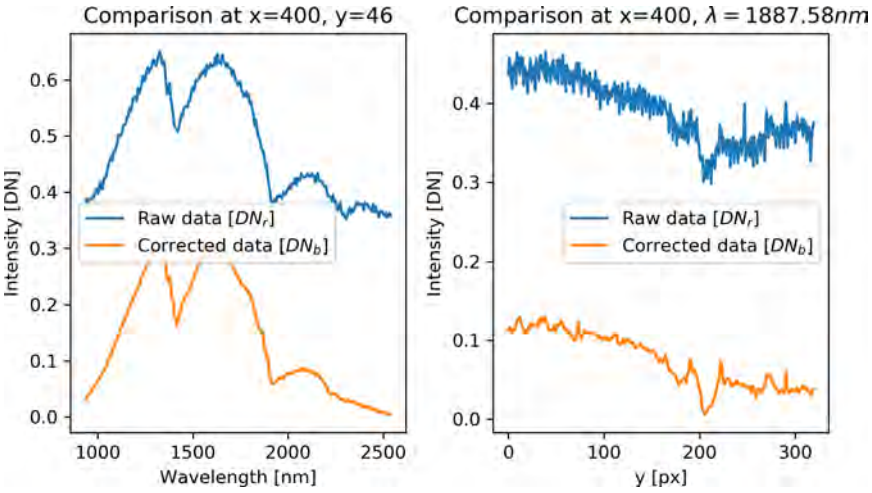


Figure 2.22: Comparison of the raw data, and the data with the computed dark frame removed.

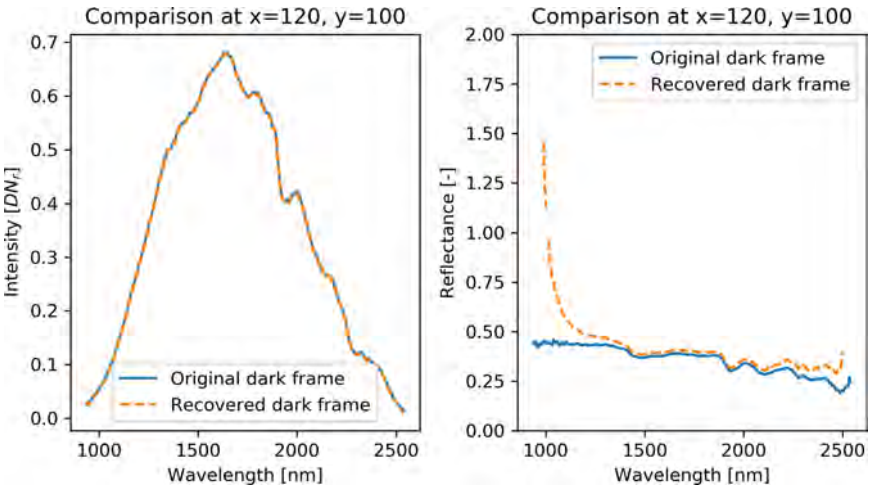


Figure 2.23: Comparison of the data computed with original dark frame, and the data with the dark frame recovered using the dark frame model in figure 2.21. The recovered data fits well the measured data. For the reflectance, the recovered data corresponds well to the original reflectance, apart from some deviation at the ends of the spectrum.

2.3 Non-Uniformity Correction (NUC)

Usually, the sensor's sensitivity is not uniform. In practice, the offset and the gain of every pixel of the sensor are independant. For any captured frame F, we have $F(y,\lambda) = k_{y,\lambda} f(y,\lambda) + m_{y,\lambda}$, where $k_{y,\lambda}$ and $m_{y,\lambda}$ are respectively the relative gain and offset of the pixel, and f the frame which would be captured if the sensor was uniform.

This non-uniformity is not usually considered when computing reflectance using full white frames, because it has no effect, as for every pixel (W is the white frame captured, D the dark frame captured, and w and d the theoretical white and dark frame respectively):

$$r = \frac{F-D}{W-D} = \frac{kf+m-kd-m}{kw+m-kd-m} = \frac{kf-kd}{kw-kd} = \frac{k(f-d)}{k(w-d)} = \frac{f-d}{w-d} \quad (2.2)$$

However, when working on spectral data, instead of reflectance, the k factor is still present:

$$S = F - D = kf + m - kd - m = k(f-d) \quad (2.3)$$

This has an impact when the white data does not cover the full sensor, as an extrapolation would not consider this factor.

Computing k is easy if it is possible to capture an uniform frame of known spectrum. This is usually done using a blackbody calibration source [41], which provides a known spectrum with very little spatial variability, however these devices are expensive and may be not readily available.

2.3.1 Computing non-uniformity correction coefficients

We propose another strategy: consider a uniform object (like a white reference), illuminated by a single uniform spectrum light source (see section 2.1.2.1). The measured frame may have intensity differences, but is spectrally uniform. Therefore, every pixel of the frame can be expressed as: $f_{y,\lambda} = I_y \cdot s_\lambda$, where I is the intensity vector, of the length of the spatial frame, and s the spectral information. If a number of frames of this type are captured, it is possible to identify a systematic bias between the theoretical frame and the measured frame, which is k.

The full problem, finding $k_{y,\lambda}$, I_y and s_λ for all y, λ which minimizes $\|k_{y,\lambda} \cdot I_y \cdot s_\lambda - F_{y,\lambda}\|$ for all measured frames F, is computationally difficult. Therefore an iterative approximation was used:

1. Set $k_{y,\lambda} = 1$, for all y, λ
2. For each frame F_i:
   (a) Find I_y and s_λ for all y, λ which minimizes $\|I_y \cdot s_\lambda - \frac{F_{y,\lambda}}{k_{y,\lambda}}\|$
   (b) Compute $I_y \cdot s_\lambda$, this corresponds to the smoothed frame: G_i.
3. Compute $\tilde{k}$, the average of the $\frac{G_i}{F_i}$.
4. Update k as the average of all previously computed $\tilde{k}$
5. Go to step 2, if the serie of $\tilde{k}$ hasn't converged yet (in practice it requires less than 5 iterations).

To minimize $\|I_y \cdot s_\lambda - \frac{F_{y,\lambda}}{k_{y,\lambda}}\|$, the constraint at every pixel can be expressed as $I_y \cdot s_\lambda = \frac{F_{y,\lambda}}{k_{y,\lambda}}$. If the logarithm is applied on both sides, we get a linear problem, which can be solved using least squares:

$$\log I_y + \log s_\lambda = \log F_{y,\lambda} - \log k_{y,\lambda} \tag{2.4}$$

2.3.2 Application example

Figure 2.24 shows the value obtained for the Specim SWIR camera. Figure 2.25 shows an example before and after the NUC correction was applied.

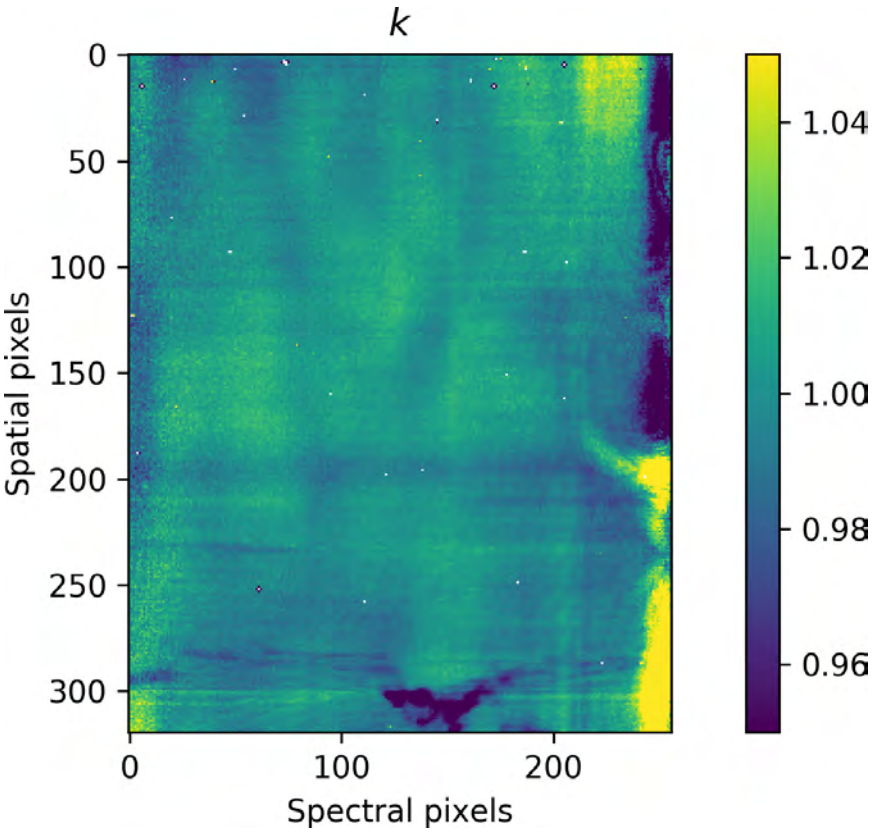


Figure 2.24: k obtained for Specim SWIR camera. Slight striping occurs due to low signal-noise ratio for high wavelengths values.

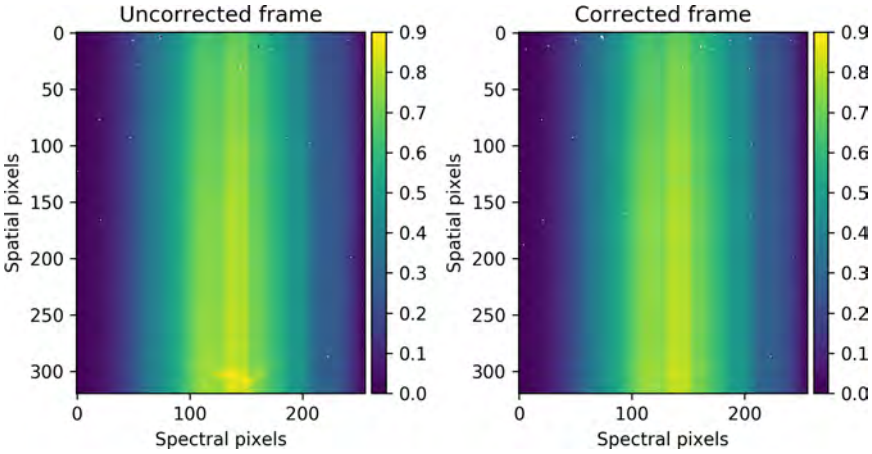


Figure 2.25: Image before and after the NUC correction using the factor k from figure 2.24. Note how the defect at the bottom is fixed by the correction.

2.4 Focus

To obtain good quality images, the image should be focused correctly. Even though methods exist to improve the quality of a blurry image (deconvolution), having the best possible optical image yields better quality.

It is quite uncommon to have a hyperspectral camera with an active focus system, therefore the method has to be passive.

A suggestion found in user manuals is to use a mire composed of black and white stripes, and to try to turn the rig in order to get a sharp image. This has proven to be impractical, for multiple reasons. In fact, putting a mire exactly on scanning target may be challenging, since it may not be flat, or not easily reachable (in a container for instance). It is also not easy for the operator to identify if the mire is in focus or not. It is also difficult for the operator to see if his actions improve the result.

On the other hand, contrast based methods, directly on subject, are commonly used in photography [42] and microscopy [43] [44]. Other approach can also be used (see [45] for a comparison).

A passive focus method has three components (see also figure 2.26):

- determination of the focus region;
- sharpness measurement;
- a peak search algorithm, to find the best sharpness.

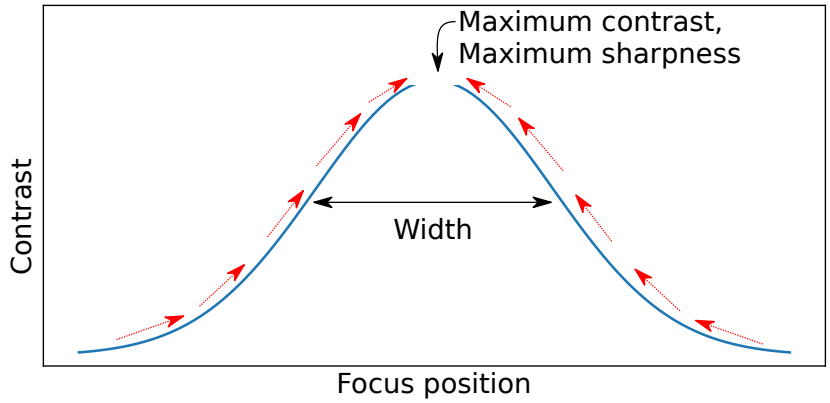


Figure 2.26: Illustration of a contrast function. The usual approach is to do hill-climbing (red arrows). The width of the constrast function is important as a very narrow peak may be difficult to locate.

Since many hyperspectral cameras do not have a motorized rig, the method has to rely on the operator. Additionally, as most are line scan cameras, it is required to have a simple algorithm which can work will a small quantity of data. Usually, considering the full frame as the focus region is not a bad choice, as the background can be choosen not to have any contrast (uniform surface).

As the method relies on the operator, there is no peak search algorithm, but the operator must be able to see the sharpness measurement in an intuitive way.

2.4.1 Optimizing focus

The method consists in giving a live plot of the last 20 seconds of a contrast measurement to the operator, which tries to increase the value by changing the focus, either by moving the subject or turning the focus rig.

Let $F(y, \lambda)$ be an image captured by the camera, and $D(y, \lambda)$ a dark frame. As a contrast measurement, multiple choices are possible. Some examples are:

- $m_1 = \frac{\sum_\lambda \sum_{y=1\ldots h-1} |F(y+1,\lambda) - F(y,\lambda)|}{\langle F(y,\lambda) \rangle_{y,\lambda}}$
- $m_2 = \sum_{y=1\ldots h-1} \frac{\left| \langle F(y+1,\lambda) \rangle_\lambda - \langle F(y,\lambda) \rangle_\lambda \right|}{\langle F(y+1,\lambda) \rangle_\lambda + \langle F(y,\lambda) \rangle_\lambda}$
- $m_3 = \frac{\sum_{y=1\ldots h-1} \left| \langle F(y+1,\lambda) \rangle_\lambda - \langle F(y,\lambda) \rangle_\lambda \right|}{\langle F(y,\lambda) \rangle_{y,\lambda}}$

Intuitively, m_1 is the sum of all difference between spatially neighbor pixels, divided by the average intensity of the frame, m_2 is the sum of the relative difference in intensity between neighboring spectra, and m_3 is the sum of the absolute difference in intensity between neighboring spectra, divided by the average intensity of the frame. It is required to normalize at some point, to limit the influence of unstable light source intensity.

Let also define $m_4 \dots m_6$ similarly, by replacing $F(y, \lambda)$ by $F(y, \lambda) - D(y, \lambda)$ in all expressions.

In the next section, each of these functions will be evaluated.

2.4.2 Application example

To compare the quality of these metrics, a sample was positionned in front of the camera. Before starting the recording, the camera was set to be way out of focus. Then, while recording, the focus rig was manually turned, reaching focus, and then continuing after till it was also very far from focus.

The contrast measurement functions were plotted as a function of the frame number. The steps are expected, since it's not possible to smoothly turn the rig manually.

The criteria to be a good contrast measurement function are:

1. its maximum should be at the focus;
2. it should be increasing while turning the rig in the right direction;
3. it should be as wide as possible, to avoid mostly flat regions, in which the operator has no feedback to know in which direction to turn the rig.

All the functions satisfy these criteria, but functions m_4, m_5 and m_6 are a bit wider. Function m_4 is the widest, and simpler computationally, and therefore was selected and implemented in application.

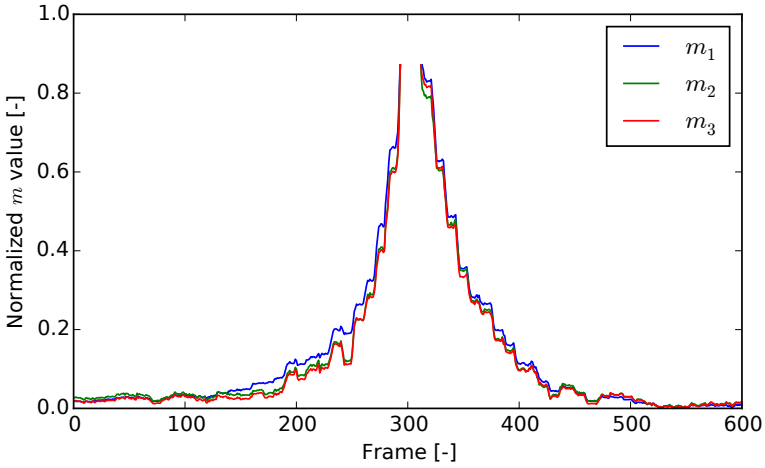


Figure 2.27: Contrast measurement function of m_1 to m_3.

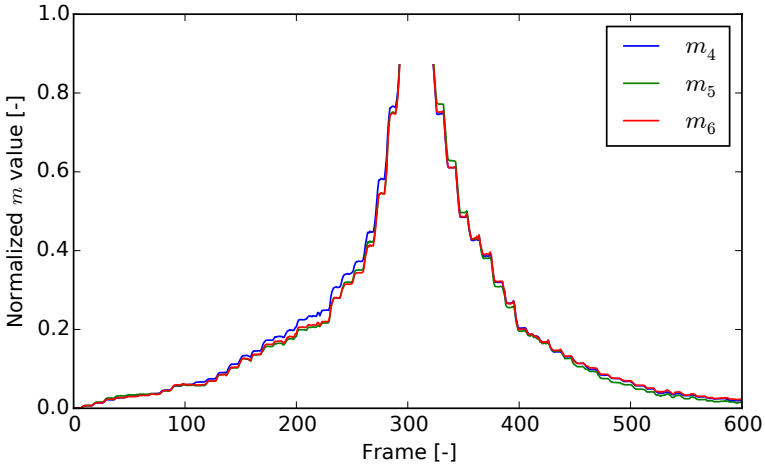


Figure 2.28: Contrast measurement function of m_4 to m_6.

2.5 High Dynamic Range imaging

It is quite common to have a scene with a very wide light intensity range, due to difference in material, illumination (shades) and geometry. The

human eye has the ability to adapt quickly to luminosity changes, in order to capture the full image without saturation or underexposure. A low exposure decreases the signal-to-noise ratio, while saturation implies data loss. In photography, the sensor has a limited dynamic range, and take a unique integration time per image. A technique named high-dynamic-range imaging consists in taking multiple images of the same scene, with different integration times, and then to merge them in order to simulate human perception [46].

Hyperspectral imaging is similar to photography, but there is no perception involved, only spectral accuracy. Moreover, saturation or underexposure can occur only at certain wavelengths. Similarly to photography HDR, it is possible to combine multiple images to improve the quality. Intuitively, the high intensity areas should use data captured at low integration time, while the low intensity areas would require higher integration time. The goal is to diminish the noise, while simultaneously avoiding saturation. An example is shown in figure 2.29.

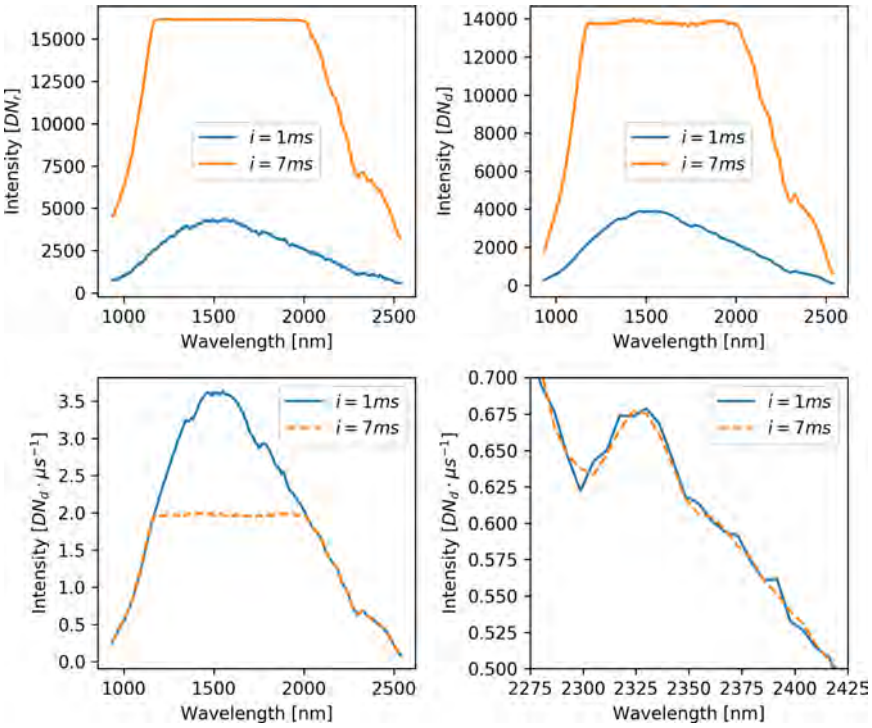


Figure 2.29: On the top left, two different raw spectra captured by the camera sensor. We can see that there is saturation on the 7 ms curve. On the top right, the same data with dark frame removed, the curves are smoother except of course in the saturated area. On the bottom left, the data was normalized by the integration time. Note how the curves are superposed, except in the saturated range. It would be tempting to only keep the unsaturated one, however the image with low integration time is also noisier, as we can see in the close up (bottom right plot).

2.5.1 Implementing HDR for improved image acquisition

First, multiple images $f_i(y, x, \lambda)$ are captured with different integration times t_i, without moving the camera.

In theory, these images should be exactly superposed, but in practice, latency issue may lead to a small shift between images. Therefore, image registration should be applied.

Two different types of image registration were applied:

- digital image correlation (DIC) using Fourier transform [47];

- feature matching using descriptors. This consist in finding significant points (for example ORB descriptors [48]) in images, and then matching them to find the geometric transformation which would superpose both images.

DIC gives very good results on signals with very low variability or when the signal size is small, but is very strict on the requirement that the image should be only translated (no rotation). On the other hand, ORB descriptors are more robust, but require well defined spatial structure (edges, corners), which is not always the case depending on the sample of interest.

Once all the images have been registered ($f'_i(y, x, \lambda)$), for each pixel, the slope of the pixel value in function of the integration time should be computed. To achieve this, the following least square system can be solved:

$$\beta_0(y, x, \lambda) + \beta_1(y, x, \lambda) t_i = f'_i(y, x, \lambda) \quad \forall y, x, \lambda, i \text{ such that } 0 \leq f'_i(y, x, \lambda) \leq \omega \tag{2.5}$$

To avoid using over-exposed pixels information, one should defined ω such that all the information is in the linear range of the CCD. For example, the pixels value of the SPECIM camera are between 0 and 16383, $\omega = 14000$.

The explanatory $\beta_1(x, y, \lambda)$ is the HDR image, and is in DN μs^{-1}.

2.5.2 Application example

As obtaining reflectance is usually the goal of hyperspectral scanning, let us see how HDR improves its measurement. Reflectance computation commonly faces two problems:

1. Saturation of some bands of the sample or of the white reference measurement, resulting in missing values.
2. High uncertainty in bands which have low illumination (as there are obtained by dividing a small value by another small value).

As an example, we captured hyperspectral images of a tourmaline sample, using the setup in figure 2.10. As it can be seen in figure 2.30, there is very large difference in luminosity throughout the sample (due to the difference in reflectance, and also to the geometry). Therefore, integrations times between 1.00ms and 15.00ms were used. As expected, the two effects previously mentioned affect the image (fig. 2.31). These effects are also visible in the reflectance spectra, but it is possible to obtain, using the HDR algorithm, a good quality reflectance both for high and for low intensity measurements (fig. 2.32).

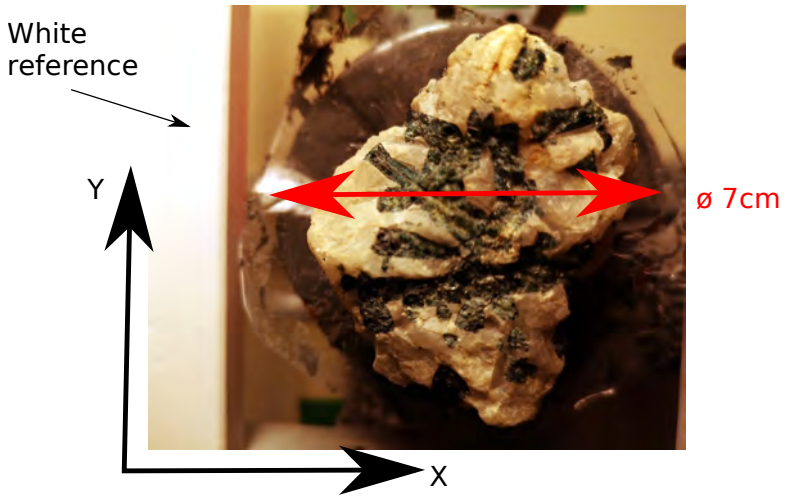


Figure 2.30: Tourmaline sample. As it can be seen on this image, the brightness is very uneven, due to the fact that the halogen lamp is on the top, and to the fact that there is an important luminosity difference between tourmaline and quarz. In fact, the brighter pixels are about one order of magnitude brighter than the darkest ones.

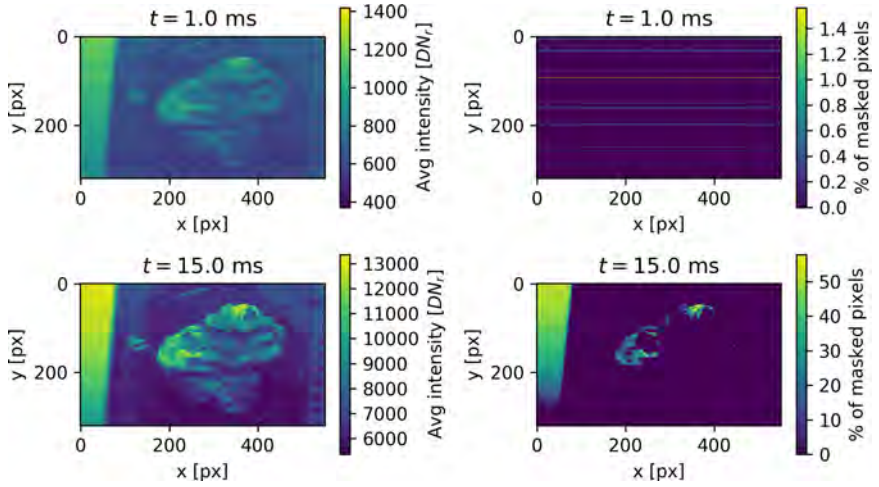


Figure 2.31: Average intensity of the raw uncalibrated image, and number of invalid (saturated or defective) pixels using integration time 1.00ms and 15.00ms. We can see that in the first case there only stripes, which are due to the bad pixels of the sensor, but the image is quite noisy, as it has a low signal to noise ratio. The second case show a much better quality image, but on the other hand some areas have more than 50% saturated bands.

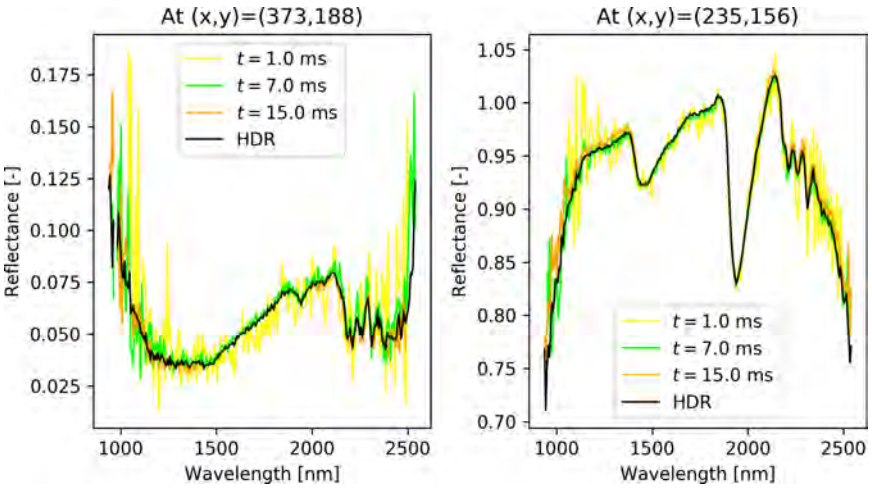


Figure 2.32: Reflectance computed using the image with integration time of 1ms, 7.00ms and 15.00ms, and the reflectance computed using the HDR image. We can see that the noise level decreases as the integration time increases. On the other hand, saturation of the white frames lead to missing data on the 15.00ms curve. We can see that the HDR algorithm is useful, since reflectance curves obtained from HDR data is very smooth, and also doesn't have missing data.

2.6 Scanning translucent objects

When scanning translucent objects, the main challenge is that the observed light may be either reflected or transmitted by the object, but can also be reflected after some penetration into the object. To simplify, we will only consider the apparent properties of the object, without taking into account where exactly the reflection took place. This simplified model considers the objects as having no depth.

2.6.1 Separating transmittance and reflectance

Capturing data from such an object requires to be able to change the light transmitted through the object. The simplest way is to be able to change the background without moving the object, and measure how the spectrum at every spatial point changed. This difference will be due to the change of the transmitted light. A schematic of such a setup is shown in figure 2.33, and a picture in figure 2.34.

We also assume that the illumination is spectrally uniform, both on the object and on the background. This can be done by having the light source

on the side. We also require that there are background pixels visible on the sides of the object. An example of an image satisfying these criteria is shown in figure 2.35.

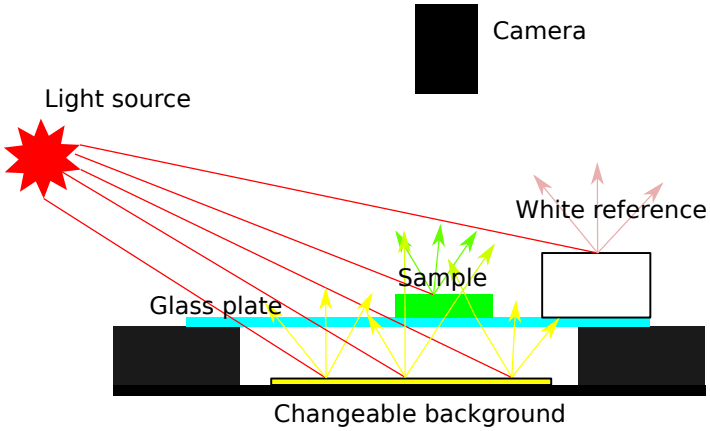


Figure 2.33: Setup to capture data from translucent objects. It is important that the elements be positioned in such a way that everything is illuminated, and that the light is able to go under the sample directly. The background should be changeable without touching the rest of the setup.

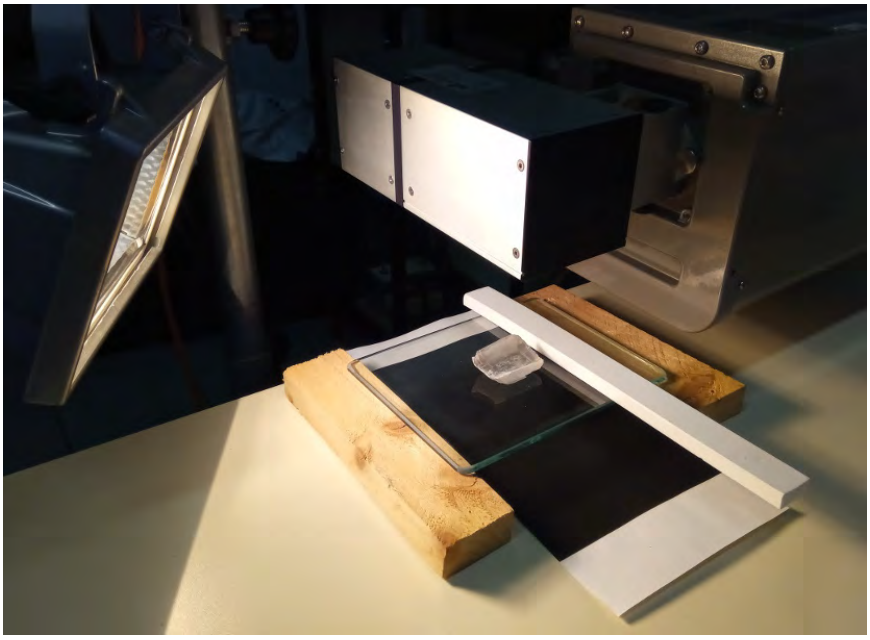


Figure 2.34: Setup to capture data from translucent objects.

Let G be the image that would be captured without the sample, I be a full white image, and F the actually captured image. The unknowns are r the reflectance and t the transmittance of the object. We have:

$$F = G \cdot t + I \cdot r \tag{2.6}$$

However, usually the images are captured as reflectance images, dividing all the data by the white data. Therefore the previous equation can be divided by I, we obtain:

$$\frac{F}{I} = \frac{G}{I} \cdot t + r \tag{2.7}$$

Since $\frac{F}{I}$ is the output of the usual scanning method to obtain reflectance, it remains to get G. One approach is to remove the sample and do an additional scan. However, it is faster to compute $\frac{G}{I}$ by doing linear interpolation of the background over the object.

By changing the background, multiple couples of $\frac{F}{I}$ and $\frac{G}{I}$ can be generated. Since there are two unknowns, at least two different backgrounds are required. Finally, the system can be solved pixel- and wavelength-wise using least squares.

2.6.2 Application example

For this example, we will capture data of a gypsum sample, using the setup in figure 2.34. The scan seen from the camera's point of view can be seen in figure 2.35.

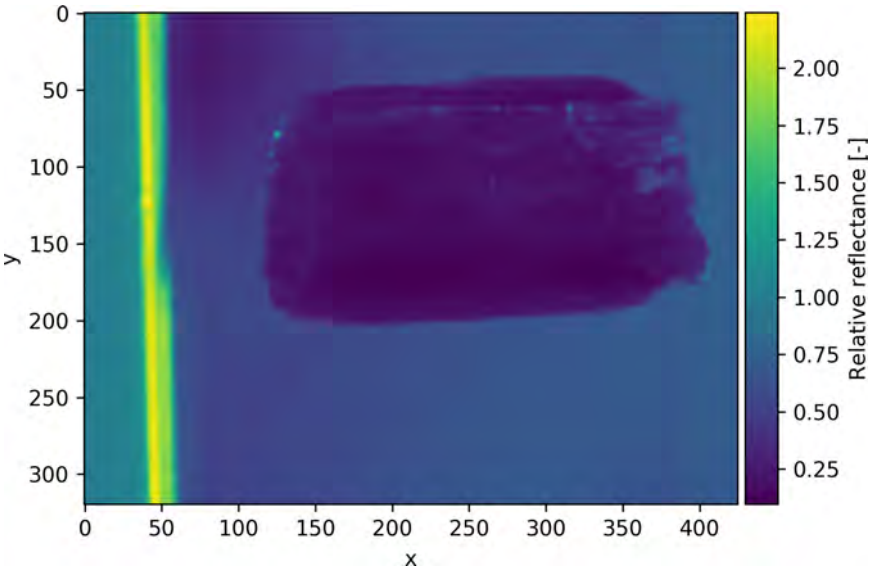


Figure 2.35: Example of a hyperspectral image, captured using the setup in figure 2.34. This is a gypsum sample in front of white paper. The pixels at $x < 40$ are the white reference. The darker area on the top left is the shade of the sample.

First, it is advisable to check if the sample is indeed translucent, as it may behave differently in infrared than in visible light. We can measure the spectrum at the same point, with two different backgrounds (see figure 2.36). As we can see, the difference is very significant. If we apply the algorithm shown previously, we obtain a transmittance and a reflectance image (fig. 2.38). As we can see, the object is mostly transparent in some wavelengths (fig. 2.38).

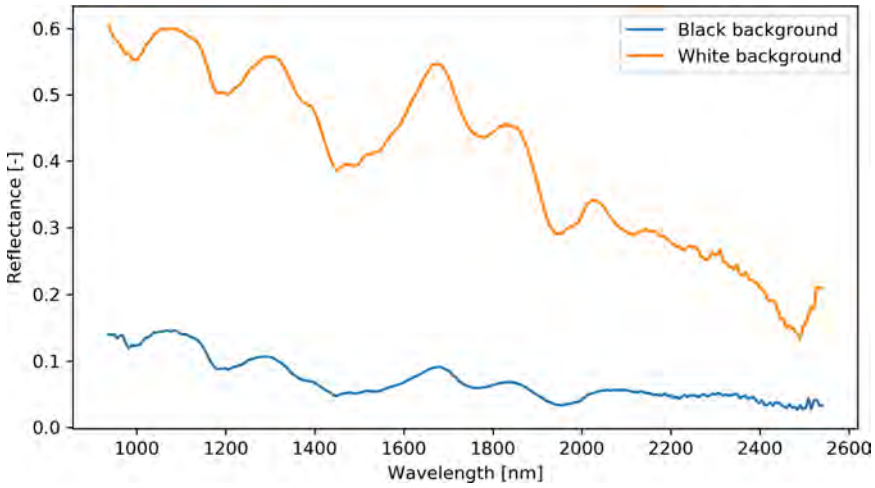


Figure 2.36: Comparison of the spectrum at one point with two different backgrounds.

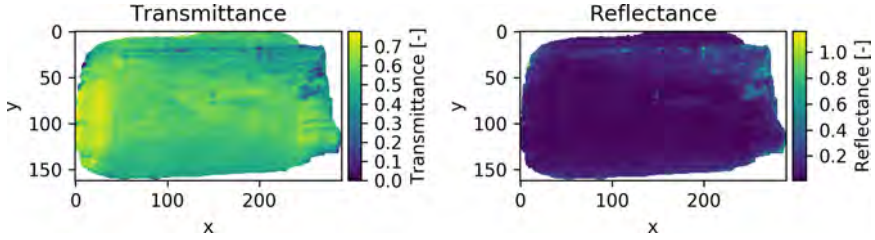


Figure 2.37: Average over all wavelengths of the reflectance and the transmittance. One can see that this object is mostly transparent.

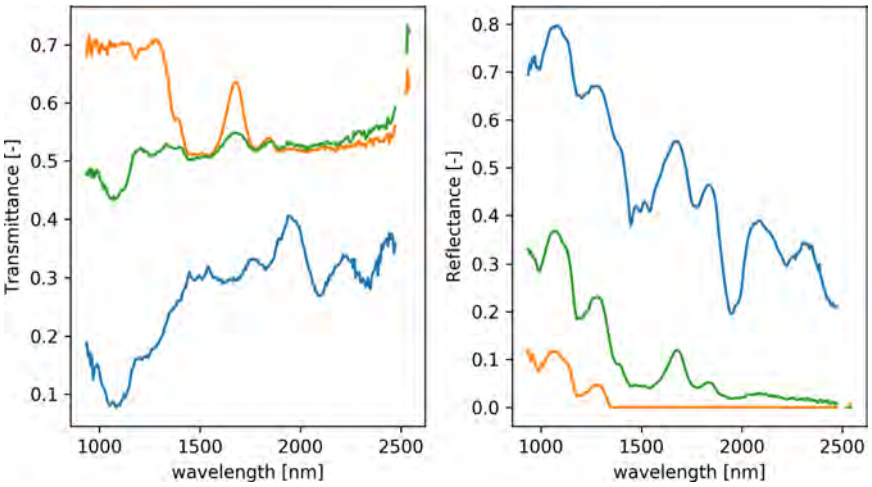


Figure 2.38: Transmittance (left) and reflectance (right) spectra of three different points in the gypsum sample. Blue is at $(267, 42)$, orange at $(98, 110)$, and green at $(158, 44)$. Detailed analysis of these spectra and the relevant phenomena is out of the scope of this article, but it is interesting to notice that for the thinnest part of the sample (orange line), there is strictly no reflectance at some wavelengths, while the transmittance is high. An attempt to scan this sample without separating transmittance would have resulted in data mostly from the background.

2.7 Conclusions

Acquiring high quality hyperspectral images is always a challenge in practice. To ensure the best possible quality and reproducibility, this paper proposes and illustrates a set of techniques allowing to overcome the most important difficulties that must be considered in practice. In particular, the paper provides practical recommendations concerning the following issues:

- How to avoid lowering the quality of the image due to bad illumination and bad scene setup
- How to limit the noise level due to dark frames, by modelling the dark frame behaviour of the camera
- How to avoid blurry images due to bad focus
- How to avoid over- and under-exposed images using HDR
- How to correct artifacts due to the non-uniformity of the camera sensor.

- How to scan translucent samples.

Using theses techniques, the paper shows how reproducible scans can be obtained if rigorous but rather simple methods are employed. In particular, improving the setup and assessing the quality of the illumination takes very little time, and has a huge impact on the quality of the results.

Moreover, it reduced the required time per scan, as an explicit method was available to devise each parameter (which were previously obtained by trial and error). Assisting the operator with the focus is faster, and results in better sharpness.

Finally, all these techniques have quite low computing requirement: a laptop is able to do everything in real time, except the HDR imaging.

If the resources available to increase the image quality are limited, we would suggest applying the techniques in the following order:

1. Improve the setup and assess the quality of illumination (this requires no special hardware nor software, and can be done in a few hours).
2. Understanding the dark frame behaviour of the camera helps to avoid basic interpretation mikstakes on the results. Only implement the interpolation proposed in section 2.2.1 if the variations are significant compared to the signal measured.
3. Use a contrast based method for focus (the algorithm to apply on frames is very simple, and both the ease of use of the camera and the quality are greatly improved).
4. HDR imaging is more complex to implement, but simplifies greatly the acquisition of data on samples which have a high variation of luminosity.

The remaining techniques are required only if the sample is translucent, or if complete white reference frames are not available.

There is still room for improvement: most of these techniques assume a linear behaviour of the sensor. While this is a good approximation, the CCD sensor is in practice not strictly linear. More advanced methods would reach higher quality, at a cost of more calibration data and more computing power requirements.

Acknowledgement

We would like to thank the Swiss National Cooperative for the Disposal of Radioactive Waste (NAGRA) for participating in the funding of this study, and Jens Becker for his support and advice.

Chapter 3

A robust input layer for neural networks for hyperspectral classification

This article presents an input layer based on the Haar transform for artificial neural networks used for hyperspectral data classification. This input layer is designed to perform efficiently with incomplete data, and is independant of the specific bands used by the camera. This enables providing pre-trained neural network, which can be used with a camera with different specification than the one used for training. This paper shows that a classifier for mineral identification built using this approach performs better than standard normalization on incomplete spectra, and similarly on complete spectra. Additionally, it shows that such a classifier matches local spectral features, and therefore that the artificial neural network is matching the spectrum shape.

3.1 Introduction

Artificial neural networks (ANN) have been widely used for various classification problems, including hyperspectral data classification. The flexibility and the performance of ANNs make them common for numerous applications such as classifying aerial images [49, 50, 51, 52], food industry [53], bacteria identification [54], nitrogen concentration estimation in rice leaves [55]. In addition to classification, neural networks can also be used for unmixing [56].

Hyperspectral images have many bands. The number of these bands and their center wavelengths depend on the camera model. Usually there is strong correlation between bands, as the spectral features are commonly wide enough to span over multiple bands. The papers previously cited rely on dimensionality reduction before the ANN. This has the advantages of reducing the size of the input data (thus reducing the requirements of the neural network) and of separating the input features. However, dimensionality reduction has some drawbacks:

- The number of dimensions kept is a choice to be made, which adds additional hyperparameters.
- Dimensionality reduction is not likely to isolate spectral features. This means that one coefficient of the output depends on the whole input spectrum.
- Dimensionality reduction techniques consist in finding the transform that can express the diversity of data using less variables. As such, a dimensionality reduction is dependant on the data for which it was computed and its parameters can be seen as additional hyperparameters of the network. Therefore, the network will have reduced efficiency if additional training or retraining is applied using different data than those that were used to compute the transform.

Due to the complex acquisition procedure of hyperspectral data, the data acquired is of imperfect quality. Neural networks are resilient to noise, but missing data is an issue. Missing data can arise for three different reasons (see fig. 3.1):

- The sensor has bad pixels, which should be masked.
- The sensor is saturated for some bands, either for the sample measurement or for the white reference. As the value of such bands is capped to the maximum possible output value of the sensor, these bands should be masked as the data is not reliable.
- Since the training is computationally demanding, pre-trained neural networks can be distributed. In that case, it can happen that the bands of the data are not the same as those that were used during the creation of the ANN, resulting in missing bands.

Handling data with missing bands is challenging in general. Imputation is commonly used to compute the missing values, usually using interpolation, replacement by the average of the data or replacement by the maximum likelihood [57]. Interpolation is however not straigthforward for missing spectral ranges, can be computationally demanding and may lead to providing bad quality input data to the ANN. The other approach available is

marginalization, which consists in ignoring bad values. It cannot be directly used with ANNs as they require a complete input layer. Imputation and marginalization can be used both with dimensionality reduction [58], or for classification algorithms [59], at the cost of added algorithmical complexity.

In this article, we present an input preprocessing method based on the Haar wavelet decomposition. This method is designed to be independant of the content of the input data set, resilient to missing data, and able to cope with different images specifications (bands center and spectral range).

The impact on the learning rate and the accuracy of the method is compared to standard normalization and is evaluated using a dataset of 73 different minerals. This dataset is presented in more detail in chapter 4 .

This paper considers only single spectrum classification, as applying spectral-spatial classification (using convolutional neural networks) would "smooth" the difference between the various methods compared.

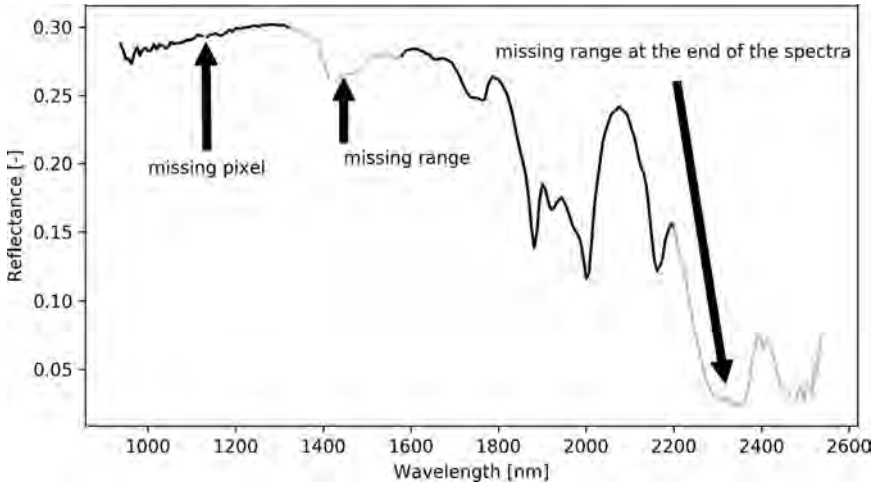


Figure 3.1: Synthetic example of missing spectral data.

3.2 Method

3.2.1 Concepts and notations

ANNs are inspired by biology, but practically they consist in a composition of linear operators and simple non-linear operators (activation function). The standard neural network is a sequence of layers, each of which is a composition of a linear operator with unknown coefficients (parameters) and an activation function. These layers link an input (source data) to an

output (the prediction), and can be visualized as a graph (fig. 3.2). The parameters of all the operators of an ANN have to be determined during a training phase. This training phase requires a lot of input data with known output prediction (ground truth), and consists in finding the coefficients which create the optimal correspondance between the output of the neural network and the ground truth. The training phase is demanding both in computing power and in the quantity of data required, as the operator corresponding to the neural network is highly unspecific. To reduce the training phase cost, it is possible to retrain only the last layer(s) of a neural network, if the classes have changed (partial retraining).

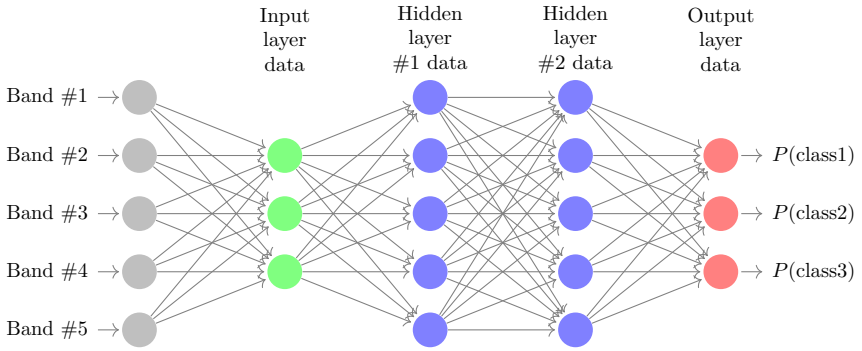


Figure 3.2: Example of an artificial neural network, with two hidden layers.

To avoid overfitting on a feature, usually dropouts are applied during the training phase. This consists in randomly dropping a fraction of the coefficients in the neural network, while compensating on the other coefficients [60]. This has the advantage that the network is "forced" to learn multiple ways to identify the same data. Dropouts can technically also be applied at the input layer.

Once the training is done, applying the neural network to the data is usually fast, as it consists of direct computations.

Throughout this document, we will work with vectors including missing values. For a given vector $\vec{v}$, we define $\vec{v}^f$ the vector where missing values are replaced by 0. We also defined $\vec{v}^m$ the mask vector, where $v_i^m = 1$ if the value v_i is missing, or 0 otherwise. Moreover, we define $\vec{v} \leq x$ to be the vector of the point-wise comparison, where each element is 1 if $v_i \leq x$ and 0 otherwise. This is a convention which is commonly used by numerical libraries.

There are some discrepancies between what terms are used to describe neural network layers depending on the context. In this document, we will use the following:

- the term *layer* is used to refer to the set of operators linking the

previous layer data to the current layer data. For example, the first hidden layer is the operators linking the input layer data to the first hidden layer data.

- the term *layer data* corresponds to the actual data vector.
- as a consequence, since normalization can be seen as a part of the network, we use the term *input layer* to refer to the operators linking the raw data to the input layer data of the neural network.

3.2.2 The Haar Inspired Input Layer (HIIL)

The input layer is inspired by the discrete wavelet transform, using Haar wavelets. Discrete wavelets transform consists in representing a function (in the present case, the measured reflectance) as a sum of orthonormal wavelets. The Haar wavelet is a piece-wise constant wavelet which makes it easy to use, and commonly used in image processing. There is abundant literature about Haar wavelets (for example [61, 62]), therefore only the specific variation used in this paper will be presented.

The transformation proposed in this paper requires 4 parameters: the minimum wavelength λ_{min}, the maximum wavelength λ_{max}, the number of subdivision levels N and the required ratio of valid values $\delta \in [0, 1]$. For example, for $\delta = 0.9$, every coefficient computed with less than 90% of the spectral range covered will be considered invalid. A missing value is considered as a 0 in the integrals.

We then proceed as follows, where r is the measured reflectance:

1. The first coefficient is the integral of the reflectance:

$$a_0 \leftarrow \int_{\lambda_{min}}^{\lambda_{max}} r d\lambda \tag{3.1}$$

2. If $N \geq 1$, for the first subdivision level we compute:

$$a_1 \leftarrow \int_{\lambda_{min}}^{\frac{\lambda_{min}+\lambda_{max}}{2}} r d\lambda - \int_{\frac{\lambda_{min}+\lambda_{max}}{2}}^{\lambda_{max}} r d\lambda \tag{3.2}$$

3. If $N \geq 2$, for the second subdivision level we compute:

$$\begin{aligned} a_2 &\leftarrow \int_{\lambda_{min}}^{\frac{\lambda_{min}+\lambda_{max}}{4}} r d\lambda - \int_{\frac{\lambda_{min}+\lambda_{max}}{4}}^{\frac{\lambda_{min}+\lambda_{max}}{2}} r d\lambda, \\ a_3 &\leftarrow \int_{\frac{\lambda_{min}+\lambda_{max}}{2}}^{3\frac{\lambda_{min}+\lambda_{max}}{4}} r d\lambda - \int_{3\frac{\lambda_{min}+\lambda_{max}}{4}}^{\lambda_{max}} r d\lambda \end{aligned} \tag{3.3}$$

4. Repeat this procedure until the number of subdivision levels has been reached. The sequence $a_0 \dots a_{2^N-1}$ is the output of the transform.

An example of this transform is shown in figure 3.3.

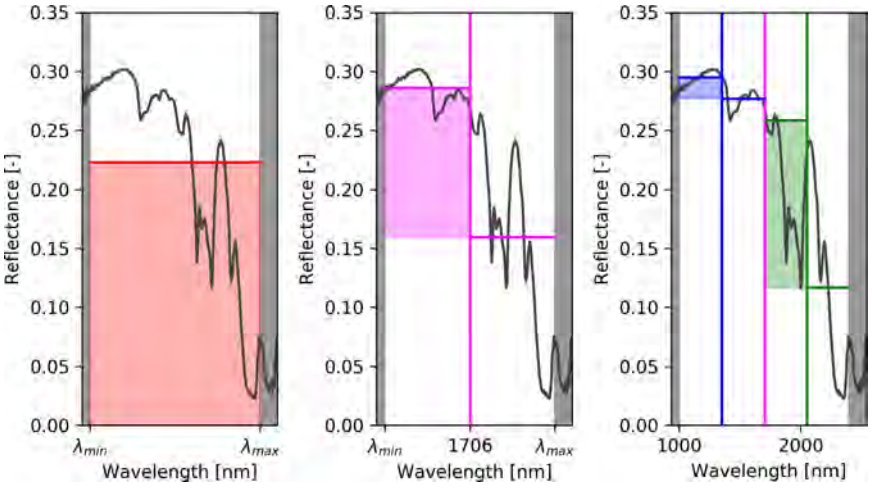


Figure 3.3: Example of Haar transform with $N = 2$ levels of subdivisions, between $\lambda_{min} = 1000\,\text{nm}$ and $\lambda_{max} = 2400\,\text{nm}$. The coefficients are $a_0 = 0.22 \cdot 1400$ (red area), $a_1 = 0.12 \cdot 700$ (pink area), $a_2 = 0.02 \cdot 350$ (blue area), $a_3 = 0.14 \cdot 350$ (green area).

In addition, set $a_i = 0$ for all i where less than δ of the spectral range is covered by actual measurements.

In practice, the reflectance output is a vector $\vec{r} \in \mathbb{R}^k$, where k is the number of bands of the camera. The physical caracteristics of the camera determines for each band the center wavelength and the width of the band. The widths of the bands can be visualized as a vector $\vec{d} \in \mathbb{R}^k$.

This transformation is a linear operator that can be discretized in a matrix $\mathbf{M} \in \mathbb{R}^{2^N-1 \times k}$. Each row contrains the coefficients required to obtain an output element a_i. The first row is constructed to cover the whole spectral range covered by the integral. Note that the end of the range can partially cover the bands, leading to coefficients $\tilde{d}_k \leq d_k$ and $\tilde{d}_l \leq d_l$. The row is structured as follows:

$$M_0 = \begin{bmatrix} 0 & \dots & 0 & \tilde{d}_k & d_{k+1} & \dots & d_{l-1} & \tilde{d}_l & 0 & \dots & 0 \end{bmatrix} \tag{3.4}$$

The next row, M_1, has both a positive and a negative range. As previously, the bands at the side can be partially covered by the integral. Moreover, the central term $\tilde{d}_c$ is a combination of the positive and negative

integral, so it can be any value between $-d_c$ and d_c, depending on where $\frac{\lambda_{min}+\lambda_{max}}{2}$ is situated in the band. The row structure is as follows:

$$M_1 = [0 \quad \dots \quad 0 \quad \tilde{d}_k \quad d_{k+1} \quad \dots \quad d_{c-1} \quad \tilde{d}_c \quad -d_{c+1} \quad \dots \quad -d_{l-1} \quad \tilde{-}d_l \quad 0 \quad \dots \quad 0] \tag{3.5}$$

The rest of the matrix is constructed likewise. A visualization can be seen in figure 3.4.

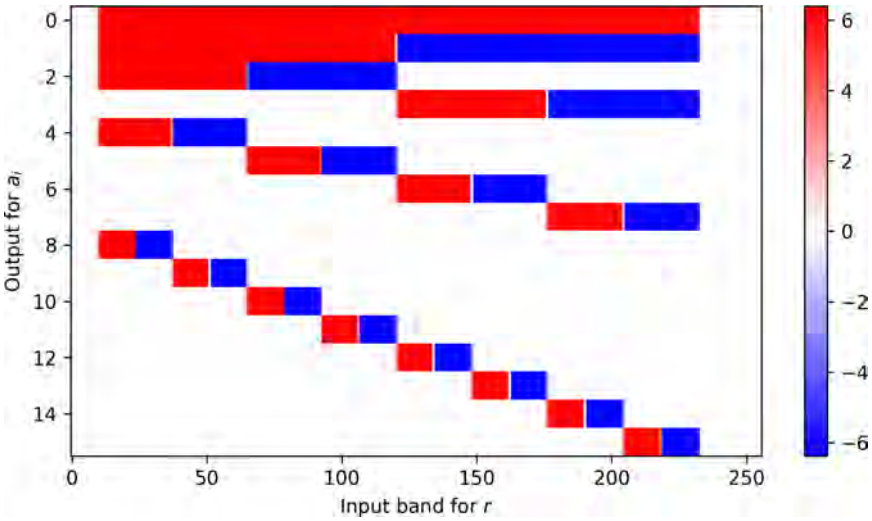


Figure 3.4: Example of matrix M with $N = 4$. The values on the side and in the center are clearly smaller due to limits of the integral not aligning on a band limit. Due to the relatively small difference in the colors, it is not possible to see that the band widths span between 6.2 nm and 6.4 nm.

Additionally, the matrix $\mathbf{N}$ is derived from $\mathbf{M}$, in which each element corresponding to a non-zero element of $\mathbf{M}$ is the inverse of the number of non-zero elements of the row in $\mathbf{M}$. Therefore, the sum of each line in $\mathbf{N}$ is 1, and it can be expressed as follows:

$$N_{i,j} = \begin{cases} 0 & \text{if } M_{i,j} = 0 \\ \frac{1}{\#\{M_{i,k} \neq 0 \forall k\}} & \text{otherwise.} \end{cases} \tag{3.6}$$

For a given reflectance vector $\vec{r}$ with missing values, we can then do the decomposition as follows:

$$\vec{a} = \left(\mathbf{M} \cdot \vec{r}^{\,f}\right) \cdot \left(\mathbf{N} \cdot \vec{r}^{\,m} \leq (1 - \delta)\right) \tag{3.7}$$

3.2.3 Evaluation of the efficiency of the Haar-Inspired Input Layer

To test that HIIL is an efficient input layer, the following needs to be checked:

- the training of the network is efficient
- the accuracy of the network is good
- the network is robust to missing data in the input

To evaluate this, a dataset consisting of 3'842'482 spectra of 73 different classes was used (fig. 3.5). Of these points, 974'544 have at least a missing band, due to bad pixels of the sensor, or due to saturation.

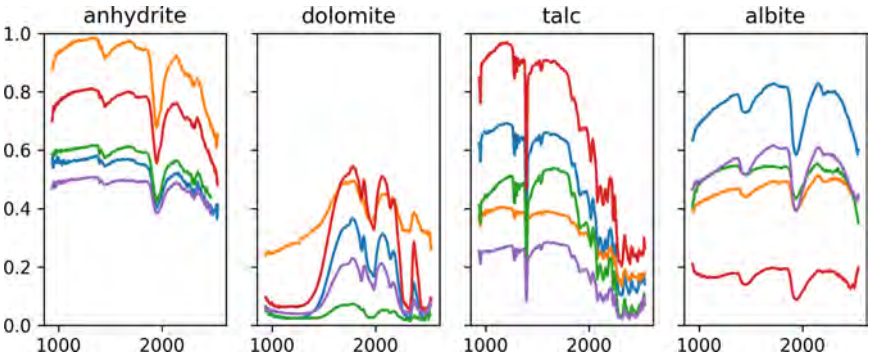


Figure 3.5: Example reflectance spectra from the dataset, for 4 different classes. For each class, 5 random spectra were selected.

The dataset was filtered to remove the incomplete spectra, and was split into three parts:

- the training dataset, containing 85% of the data points,
- the validation dataset, containing 10% of the data points,
- the test dataset, containing 5% of the data points.

Additionally, a second test dataset was created, containing 5% of the points of the original (non-filtered) dataset, of which 30% of the spectral data had been erased.

For all these steps, we used a neural network consisting of three dense hidden layers, each consisting of a linear operator, a bias, and a ReLU, of sizes 1024, 1024, 256. This is a commonly used setup [18].

Having a too small ANN size can constrain the learning, as there would not be enough coefficients to fully model the relation between the input and the output. Therefore, the size of the ANN was chosen to be large enough, as the goal is the evaluation of the influence of the input layer. In practice a smaller network size should be choosen in order to increase the computational efficiency.

The training was performed using an Adam optimizer [63]. The training data was balanced between classes, and the accuracy of the classifier on the validation dataset was stored after each epoch.

As input layers, we used:

- The identity operator, which consists in using directly the raw data as the input layer. This is known to create optimization problems during the training phase, but is useful as a reference point.
- The standard normalization. For a given reflectance data r, we use as an input layer: $\frac{r-\text{Mean}(r)}{\text{Std}(r)}$.
- HIIL, with two different transform depths $N = 6$ and $N = 8$

To avoid over-fitting, dropouts were added at the input layer, and at each hidden layer.

For each of the input layers, we evaluated the accuracy after the classifier after 1'000 epochs, for various drop-out parameters.

The goal of HIIL is to be used on data with missing values, so it needs to be evaluated on the second test dataset. However, since the other input layers require a full spectrum, we evaluated them with the spectrum linearly interpolated. This is not an optimal approach, but the more advanced approaches would require statistics about the input data, which may not be available. Moreover, this approach is computationally demanding. We also evaluated HIIL in these conditions, to see how interpolated values impact the accuracy of the classifier.

The classifier using the HIIL input layer was also evaluated on partial spectra. The principle consists in choosing a center band, a width of the interval and to mark every other bands as invalid. The accuracy of the prediction can then be computed. This can be done for every interval and every class.

3.3 Results

To assess the efficiency of the training, we compared the evolution of the accuracy on the validation dataset throughout learning (figure 3.6). We can observe that the identity input layer limits accuracy (it reaches a plateau), but other input layers yield good results, even if HIIL has a slower learning rate.

By using the first test set for each input layer while varying the dropout, we can observe that the results are similar between the HIIL and the centered and scaled normalization (table 3.1). The dropout after the input layer has a important negative effect on the training, but HIIL is less sensitive to it.

As we can see in table 3.2, the accuracy using the HIIL input layer is better than when using other input layers, even with the invalid values replaced by 0.

For partial spectra, the results for calcite, gypsum and quartz can be seen in figures 3.7, 3.8 and 3.9 respectively. Observe that classification accuracy is good as soon as the specific spectral features of the mineral are given to the classifier. For classes with nearly no specific spectral features like quartz (fig. 3.9), nearly the full spectrum is required in order to classify the spectrum correctly. Intuitively, this is due to the classifier having to ensure that each feature of the training dataset is absent.

The impact of the HIIL input layer on computation requirements to train and use the neural network was negligible.

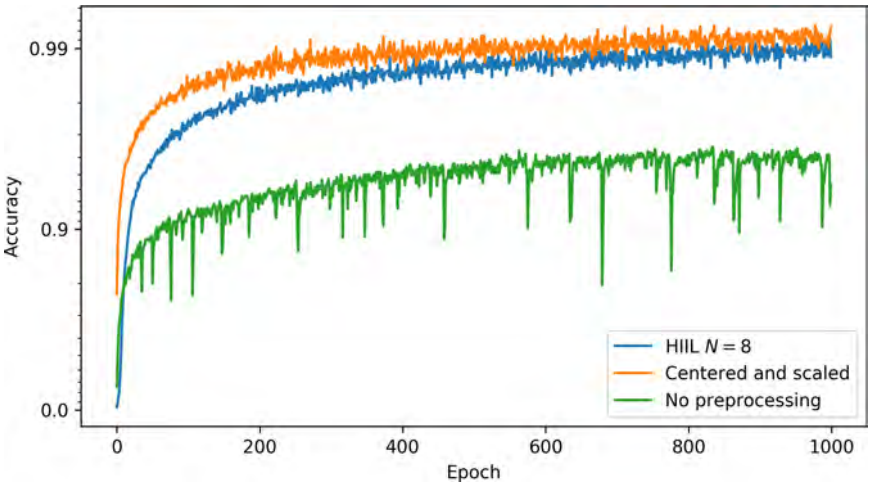


Figure 3.6: Accuracy of the ANN on the validation set. The input layer dropout is disabled, and the dropout after each layer was set to 0.25.

Table 3.1: Comparison of the accuracy on the first test set using various input layers, with layer dropout 0.00, 0.25, 0.50.

Input layer	Input dropout = 0.0	Input dropout = 0.25	Input dropout = 0.50
Identity	97.0%, 94.5%, 95.5%	53.0%, 35.6%, 37.6%	6.0%, 16.2%, 20.3%
Centered and scaled	**99.0%, 99.2%, 98.9%**	95.3%, 94.3%, 93.3%	64.7%, 80.9%, 87.1%
HIIL ($N = 6$)	98.6%, 98.6%, 98.0%	97.2%, 96.5%, 95.4%	90.2%, 87.5%, 86.8%
HIIL ($N = 8$)	98.9%, 99.0%, 98.4%	**98.4%, 97.6%, 96.7%**	**93.2%, 92.3%, 91.6%**

Table 3.2: Comparison of the accuracy on the second test set (with missing values) of the various input layers, with layer dropout 0.00, 0.25, 0.50.

Input layer	Input dropout = 0.0	Input dropout = 0.25	Input dropout = 0.50
Identity (interpolated)	61.6%, 64.7%, 67.5%	31.9%, 28.0%, 31.6%	5.0%, 14.2%, 17.1%
Centered and scaled (interpolated)	59.2%, 62.9%, 62.0%	57.5%, 57.7%, 58.0%	35.2%, 49.7%, 53.0%
HIIL ($N = 6$, interpolated)	57.2%, 60.1%, 61.5%	75.4%, 77.3%, 74.7%	68.9%, 67.2%, 66.7%
HIIL ($N = 8$, interpolated)	59.6%, 63.9%, 62.9%	74.6%, 75.5%, 73.9%	71.9%, 72.2%, 71.9%
HIIL ($N = 6$)	13.4%, 16.9%, 15.6%	64.8%, 65.5%, 64.0%	70.0%, 69.0%, 67.2%
HIIL ($N = 8$)	16.2%, 18.4%, 19.0%	66.1%, 67.1%, 65.2%	69.2%, 70.4%, 69.6%

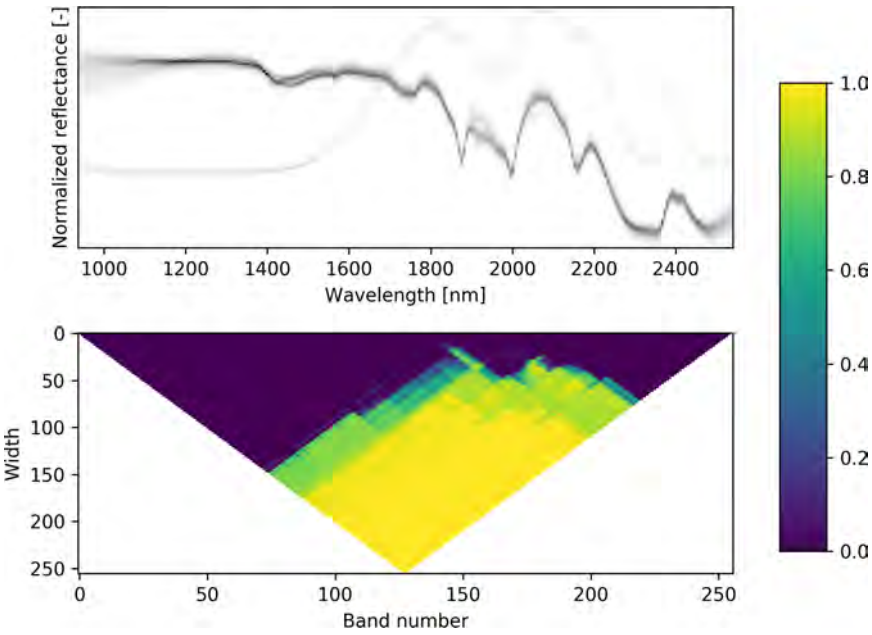


Figure 3.7: Accuracy of classification of calcite spectra, depending on the spectral area which is provided to the ANN. For example, we can see that with an interval of 50 bands around band 150, the accuracy of the classifier is around 80%. Observe that as soon as the wavelengths from the farest part of the spectra are covered, we get good recognition. This makes sense since this is the part of the spectrum in which there are variations.

3.4 Discussion & conclusions

Our results showed that HIIL is an efficient and robust input layer.

Complete spectral data was classified correctly 98% of the time. Spectral data with 30% of the data removed was still classified correctly 70% of the

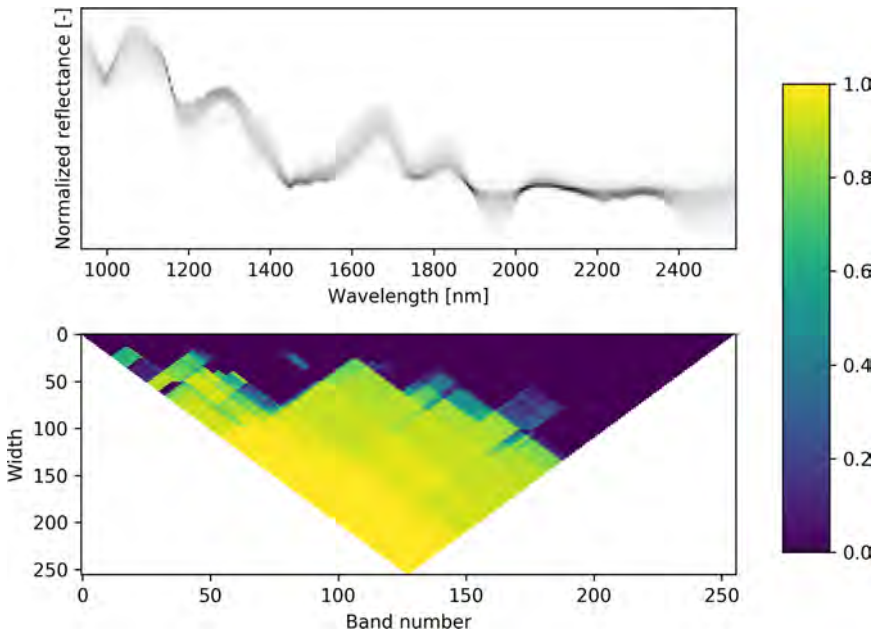


Figure 3.8: Accuracy of classification of gypsum spectra, depending on the spectral area which is provided to the ANN. Similarly to figure 3.7, we can observe that the relevant part of the spectrum is the lowest wavelengths.

time. We can observe that the input layer dropout increases the accuracy of the classifier. We also showed that a classifier built using the HIIL has the ability to find and match local features.

As expected, the presented input layer has low computational requirement, therefore it can be easily applied to practical problems. An unexpected result is that HIIL performs better if the data is linearly interpolated to fill holes.

HIIL is focusing on local features instead of global ones, unlike the methods relying on dimensionality reduction. This creates a clear relationship between specific spectral regions and neuron activation, which is more intuitive than to have activation based on a linear transform of the whole spectrum. Compared to the classical normalization approach, we showed that it performs similarly if the data contains no missing data, and performs better with real data with missing bands. It is also much simpler to implement than alternative approaches.

The high classification accuracy using HIIL is due to the following reasons:

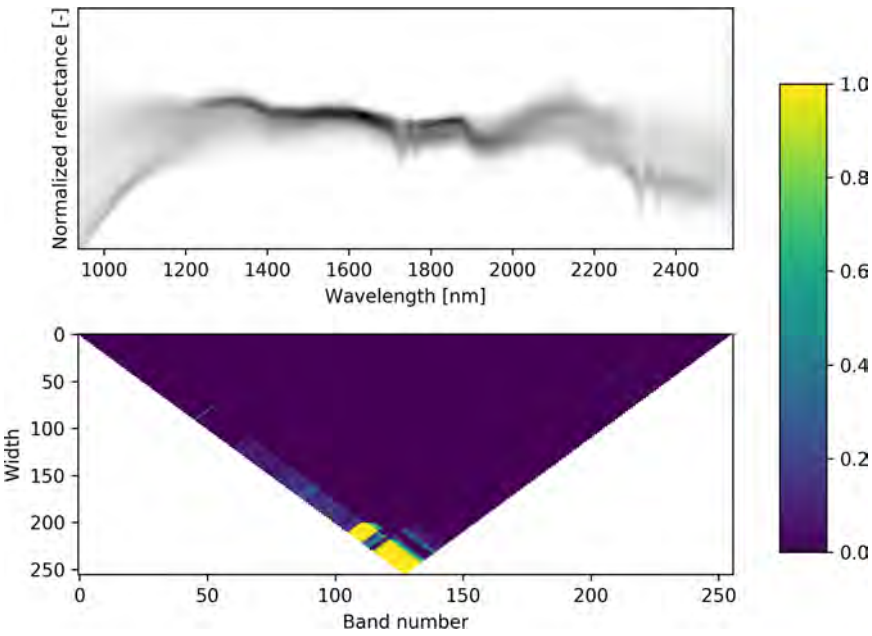


Figure 3.9: Accuracy of classification of quartz spectra, depending on the spectral area which is provided to the ANN. As quartz has nearly no spectral variations in the wavelengths considered, nearly all the spectrum is required to have a correct classification.

- The HIIL's output is 0-centered, which is compatible with the classical assumptions of ANNs layers (activation functions and dropouts)
- Missing input data generates output similar to dropout at the input layer. Therefore the network is in similar conditions during the training and prediction phases.
- The input layer data scale is proportional to the spectral range covered by the integrals. This tends to naturally increase the influence of large spectral features compared to small ones (which could be noise), therefore making the ANN focus on general tendancy instead of noise.
- The transform is independant of the input data, as it is applied individually on each spectra.

Finally, the low computational impact of the HIIL is due to the fact that its linear nature is compatible with the kind of operators used in ANNs, and therefore benefits from the optimization usually applied in the ANN software libraries to speed up computations.

The increase of accuracy when HIIL is applied with linearly interpolated data suggests that some information is not captured by the neural network. This suggests that tuning would be required, for example by changing the ANN parameters, or by changing δ.

One limitation of this study is that it does not consider the impact of using this input layer on the size and the complexity required of the ANN. It was considered not necessary because the computational requirement was in any case quite low. Moreover, dimensionality reduction techniques were not evaluated, as numerous variant exists. Besides, the algorithmic complexity of these methods is higher and the ANN requirements are quite different. We also did not consider the re-training abilities of classifiers using the HIIL, although we expect them to be good.

Further studies should be done on using different type of wavelets, or to use different transform specifications. One should also compare how this input layer performs compared to the other approaches when the camera used to train the network is not the same as the one used for prediction. Comparing its efficiency using a classifier that also uses spatial features, such as a convolutional neural network, might also provide interesting results.

HIIL is likely to be also applicable to other types of continuous data with missing parts.

To conclude, a classifier for hyperspectral mineral identification built using the Haar Inspired Input Layer has better accuracy than a classifier using standard normalization technique on incomplete spectra, and similar accuracy on complete spectra. As it has lower computational requirements, it should be considered for hyperspectral classification using artificial neural networks.

3.5 Computer code availability

Although this article present an algorithm (not an implementation), an example implementation based on `Keras` is available at `https://github.com/UniNE-CHYN/HIIL-keras`.

Chapter 4

A 2D hyperspectral library of mineral reflectance, from 900 to 2500nm

Mineral identification using machine learning requires a significant amount of training data. We have built a library of 2D hyperspectral images of minerals. The library contains reflectance images of 130 samples, of 76 distinct minerals, totalling more than 3.9 millions data points.

In order to produce this dataset, various well-characterized mineral samples were scanned, using a SPECIM Short Wave Infrared (SWIR) camera, which captures data from 900 to 2500 nm. Minerals were selected to represent all the mineral classes and the most common mineral occurrences.

For each sample, the following data is provided:

- *At least one hyperspectral image of the sample, consisting in 256 wavelengths between 900 and 2500nm. The raw capture data, the high dynamic range (HDR) image, and the masked HDR image are provided for each scan (each of them in HDF5 format).*
- *A text file describing the sample, designed to be easy to analyze by software.*
- *As supplementary information, RGB images (JPEG files) and automated 3D reconstruction (Stanford Triangle PLY files) are provided. This data in intended to help the user of the dataset to visualize and understand specific sample characteristics.*

2D hyperspectral images were produced for each mineral, which consist of many different spectra with high diversity. These are shown to contain similar spectra to the ones contained in already established spectral libraries. An artificial neural network was trained to assess sufficient quality of the dataset.

This data is mainly aimed at training machine learning algorithms, such as neural networks, but can be also used as validation data for other types of classification algorithms.

4.1 Background & Summary

One of the key applications of hyperspectral imaging is classification. In particular, various algorithms exist to determine a mineralogical map based on a hyperspectral image [64]. These algorithms require a spectral library compiling reference spectra.

The USGS Spectral library [65] is commonly used for this purpose. It provides a reflectance spectrum for a wide range of minerals, usually in powder form. This library was build using high quality spectrometers. This type of measurement provides a single homogeneized measurement per sample. This type of library is designed for standard classification algorithms such as the spectral angle mapper [10] but is not ideal for machine learning algorithms.

Indeed, machine learning algorithms such as artificial neural networks became very popular. This approach requires a large amount of data per class for the training. The training dataset should be as diverse as possible, to enable the learning algorithm to generalize as much as possible. This new approach requires a new type of spectral library, which contains as much data as possible for each class.

The dataset we propose is designed to be this new type of spectral library. We use a hyperspectral camera to produce 2D images, which contains many datapoints, contrary to libraries produced by spectrometers. This diversity of the spectra measured is achieved by avoiding homogenization, by measuring mineral samples instead of powders.

Minerals were selected from the collection of the University of Neuchâtel, and represent well all the mineral classes and of the most commonly occurring minerals [66].

4.2 Methods

This dataset contains scans of rock samples of the mineral collection. When possible, different rock samples of the same mineral were selected to ensure diversity, by having different external characteristics (such as habit or color)

of the same mineral. When multiple fragments of the same rock sample were available, data acquisition was performed on each of them.

The dataset consists in three different kinds of data, each of which is described in a different subsection:

- Hyperspectral data
- 3D reconstruction and RGB images of the sample
- Text description (provides the mineral characterization)

The hyperspectral and photogrammetry data acquisition is entirely automatic, using the setup shown in figure 4.1. The sample is placed on a 3D-printed container containing modelling clay, in order to have a mostly flat surface above. The hyperspectral camera then acquires the hyperspectral data (details presented in the following section), followed by the capture of the visible images for the photogrammetry.

Throught this section, we will often refer to data stored in files. To increase the readability, we use the following notations:

- `.extension` means the file ending with `.extension` which corresponds to the sample
- `.scan.h5[/wavelengths]` is the data at `/wavelengths` in the file ending with `.scan.h5` which corresponds to the sample.

Hyperspectral scanning and processing

The hyperspectral data acquisition was performed using a Specim SWIR hyperspectral camera. This device consists of a push-broom camera, and a mirror scanner.

The data captured should be as independant as possible of the sensor and illumination characteristics. This specific sensor return for each pixel values between 0 and 16'383, depending on the radiance of the sample at the specific wavelength. In practice, the value varies between a base noise level, up to a point where the sensor exhibits saturation. Between these points, the sensor value is linearly dependent on the radiance of the sample. To get reflectance, we need to subtract from each captured frame the base noise level (dark frame, which we obtain by capturing an image with the shutter closed), and then divide the radiance data captured from the sample by the radiance data captured on a white target (white frames). It should be noted that some pixels don't follow these rules (either are stuck on a given value or behave randomly). These pixels are considered as "bad pixels" and should be removed from the data, as the reflectance computed would be inconclusive.

Due to the geometry and the diversity of the samples, it is often not possible to find an integration time which simultaneously doesn't overexpose some pixels, and underexposes others. Therefore, to get an image without saturation, a high dynamic range (HDR) approach was used, by capturing images at multiple different integration times $t_0, \ldots, t_n$, and then merging the images. The integration times can be seen in `.scan.h5[/integration_times]`.

The following was captured:

for $k \leftarrow 0 \ldots n$ **do** ▷ Capture a white frame for all integration time
    Camera integration time $\leftarrow t_k$
    `.scan.h5[/white- k-d0]` ← DARK FRAME
    `.scan.h5[/white- k]` ← 25 WHITE FRAMES
    `.scan.h5[/white- k-d1]` ← DARK FRAME
end for

for $k \leftarrow 0 \ldots n$ **do** ▷ Capture data for all integration time
    Camera integration time $\leftarrow t_k$
    `.scan.h5[/data- k-d0]` ← DARK FRAME
    `.scan.h5[/data- k]` ← CAPTURE DATA
    `.scan.h5[/data- k-d1]` ← DARK FRAME
end for

Dark frame subtraction, bad pixel removal, and saturated value removal was then applied to `.scan.h5[/white- k]` (obtaining W_k) and to `.scan.h5[/data- k]` (obtaining D_k).

Dark frame subtraction was performed using a linear interpolation between the dark frame captured before and after the data acquisition, and was subtracted to the corresponding data. The sensor also has some bad pixels that should be removed. They were detected using the white frame: pixels either "stuck" or exhibiting large variance over time were masked. Finally, any pixel value greater than 14'000 (out of 16'383) was masked, as this was observed to be the maximum value where the sensor was behaving linearly.

The reflectance was then computed:

$$R_k = \frac{D_k}{W_k} \tag{4.1}$$

Finally, the HDR image was computed as the average of the non-masked values of the reflectance images:

$$\texttt{.hdr.h5[/hdr]} = \frac{1}{n+1} \sum_{k=0} n \frac{R_k}{k} \tag{4.2}$$

To isolate the sample from the background, a mask was constructed and applied to the data. The result was stored in `.mhdr.h5[/hdr]`.

All the files contain also a `/wavelengths` array, which contains the list of band center, in nanometers.

3D reconstruction

This dataset contains 3D reconstruction as supplementary information for the user of the dataset. These 3D reconstructions can be useful to the user of the dataset to understand specific aspects of the samples (for example, why a specific pixel has a different spectrum than the rest of the sample). The process was designed to be fully automated, and to produce satisfactory results for most of the samples. For some of them however there are issues which create problems in the 3D reconstruction (such as holes, texture or shape artifacts), which was not corrected since this data is not the main focus of the dataset,

The process is the following:

1: **for** $\alpha \leftarrow 000, 010, 020 \ldots 360$ **do** ▷ Capture an image every 10 degrees
2: Angle of the sample $\leftarrow \alpha$
3: **for** $k \leftarrow 1, 2, \ldots 6$ **do** ▷ Capture 6 frames
4: $I_k \leftarrow$ WEBCAM FRAME
5: **end for**
6: `-im-α .jpg` $\leftarrow \frac{1}{6} \sum_{k=1}^{6} I_k$
7: **end for**
8: **for** $\alpha \leftarrow 000, 010, 020 \ldots 360$ **do** ▷ Compute smoothed image
9: $G_\alpha \leftarrow$ GAUSSIAN FILTER(`-im-α .jpg`)
10: **end for**

For each G_α, an image M_α is constructed by keeping only the pixels in the convex hull of the pixels which have changed between G_α and $G_{\alpha+10}$. This in effect removes the non-moving background behind the sample. The other pixels are replaced with black pixels.

The process is then the classical structure from motion 3D reconstruction [67], with the exception that the features matching and the mapping is applied on M_α images. The other steps are performed with the `-im-α .jpg` images.

Mineral characterization

Samples were characterized for their mineralogical class, crystallographic system, habit, luster, transparency, presence of cleavage, twinning and for their color. Table 4.2 includes the terms used to characterize the samples. Not all the terms are used in the current library, but they are still listed to indicate that they were considered while describing the sample, and to allow further extension of the library. They represent a compilation of commonly used terminology in mineralogy [66], in view of future developments of the

library. Additional information (i.e. chemical composition, provenance of the sample) are provided in the `description.txt` file. Mineralogical class and crystallographic system are well defined terms, as they relate to the internal order of crystals. Morphological crystallography (i.e. external form of minerals) incurs however some degree of subjectivity. Particularly for the description of the habit, different terms can be used to describe similar characteristics. In principle, the term euhedral was preferred when describing a crystal with well-formed faces but for minerals of the isometric system, the term cubes was preferred for habit description. At the same time, a mineral can be euhedral and prismatic when describing a crystal with four or more sides of similar length and width and being elongated in one direction. The term anhedral was used for mineral presenting no development of faces, microcristalline, massive or granular for aggregates of minerals with no visible macroscopic cristallographic characteristics, and amorphous for minerals lacking an internal atomic arrangement (i.e. opale). Habit terms such as lamellar, foliated, micaceous, bladed or tabular were used to describe mineral aggregate composed of scales or lamellae. The terms of fibrous, stellated, globular, botryoidal, reniform, mammillary, colloform apply to parallel or radiating groups of individual crystals, while the terms of acicular and filiform for minerals in isolated or distinct crystals. Luster and transparency definitions are included in [66].

Code availability

The data acquisition and the hyperspectral data processing was performed using software developed inhouse. The software developed is available on GitHub: `https://github.com/unine-chyn/hics`

As the software setup for acquisition depends on the configuration and the hardware used, only an enumeration of the modules used is given. Three different classes of modules where used. Hardware related modules are specific to the hardware setup, and are likely to be modified with a different setup. Preview and control modules are used by the operator to control the camera and check that the process is performing satisfactorily. Finally, the plugins used for capturing data contain the algorithmic logic of the capture.

- Hardware related: `hics.hardware.specimswir.camera` (camera control), `hics.hardware.specimswir.scanner` (mirror scanner control), `hics.hardware.specimswir.framegrabber` (frame grabber), `hics.hardware.ev3.ev3focus` (LEGO focus control), `hics.hardware.ev3.ev3rotater` (LEGO rotating device), `hics.hardware.webcam` (webcam capture)
- Preview and control: `hics.daemon.frameconverter` (basic correction for preview), `hics.gui` (graphical user interface)

- Plugins for capturing data: `hics.plugin.photogrammetry` (capture data from the webcam), `hics.plugin.record` (record hyperspectral data), `hics.plugin.autofocus` (find the focus maximizing contrast), `hics.plugin.labelprint` (print the label for the sample), `hics.plugin.autoscan` (automate all steps required for the scan, using the various plugins)

Once data is captured, the following commands were used (`@` replaces the scan name):

```
python3 -m hics.datafile.calibrate --input @.scan --output @.
    ↪ calibrated --max 14000
python3 -m hics.datafile.merge --input @.calibrated --output @.
    ↪ hdr --clean --required_images 2
# A mask should be generated by the user in @.mask.png
python3 -m hics.datafile.maskdata --input @.hdr --mask @.mask.
    ↪ png --output @.mhdr
```

For the 3D reconstruction, after multiple experiments with various tools, it was found that the best output quality was obtained using COLMAP [67] for the keypoints detection and OpenMVS [68] for the construction of the pointcloud, the mesh, and the texturing. A small change was applied to COLMAP in order to orient the sample correctly. This change however doesn't affect the quality of the result. OpenMVS was used without any modification.

The code to automate the build of the tools and the processing of the images is available at `https://github.com/unine-chyn/sfm-bundle`.

4.3 Data Records

The data is published on Zenodo as 3 different data records, depending on which hyperspectral data is included:

(Data Citation 1) contains the raw scan data

(Data Citation 2) contains the HDR data

(Data Citation 3) contains the masked HDR data

Each data record is composed of a zip file for each sample, named `####.zip`, where `####` is the ID of the sample as available in table 4.1. The data records also contain the ENVI template (provided in the file `envi_template.xml`.

Each `zip` file contains the text description file: `description.txt`. It also contains the data of one or two measurement, `A` and possibly `B`. The files are named as follows (`####` is the sample ID, `@` the measurement letter):

- `####-@.ply` and `####-@.png`: 3D reconstruction in Stanford PLY format, and associated texture file.
- `@-im-000.jpg`, `@-im-010.jpg`, ..., `@-im-360.jpg`: JPEG images of the sample, the angle is in degrees. 0 degree correspond to the position used during scanning.
- `@.scan.h5`, `@.hdr.h5`, or `@.mhdr.h5`: hyperspectral data in HDF5 for (Data Citations 1), (Data Citations 2), and (Data Citations 3) respectively.

4.4 Technical Validation

It is difficult to compare different spectral libraries directly, because they have different samples and different sample preparation technique. However, the spectra measured have similar shapes than the corresponding ones in the USGS library. For some samples, very similar spectra can even be found (fig. 4.2).

It is difficult to assess that spectra are "correct", as the measured spectrum depends on the sample. As the samples of the same mineral in different libraries are different, we cannot expect them to have the same spectra and therefore use this method to ensure validity of our library. Instead, we can test that the measured spectra are specific to the sample class tested. To this mean, a common approach is to split the dataset, to have a classifier learn on one part of it, and to check the rate of correct classification on the part that was not used for learning. A efficient classification will indicate that the classes are well separated, while a bad classification accuracy will indicate that either the classes are not well separated, or the classifier was not efficient. A inefficient classifier could be due to either insufficient or incorrect training, or for it to be not able to capture the complexity of the data. Therefore, it is sufficient to show satisfactory classification efficiency for a given method (as the data size prevents a classifier to overperform by "luck").

We followed this approach this approach by splitting randomly the hyperspectral dataset into three parts:

- the training dataset, containing 85% of the data points,
- the validation dataset, containing 10% of the data points,
- the test dataset, containing 5% of the data points.

The validation dataset is used to assess that the classifier is learning correctly, but the efficiency is checked on the test dataset after the learning

is completed. Multiple different stuctures of artificial neural networks were trained in order to check how well the sample were identified.

The accuracy, while it depends of the classifier used, peaked at 98%. This is likely to be due to the data. The simplest one to achieve such correct classification rate is a multilayer perceptron of layer size 1024, 1024 and 256, with scaled input (by subtracting the mean and dividing by the standard deviation). The 98% of correct classification on the test dataset suggests that the dataset is of good quality. Most of the 2% wrong classifications are on native elements and oxides samples, which mostly reflect light and don't have well defined reflectance spectra.

4.5 Usage Notes

To ease the re-use and the understanding of the image, a template is provided along the data files to load the data in ENVI. For `.scan.h5` files, the template "fasnacht-hicsdata (SCAN)" should be used. For `.hdr.h5` and `.mhdr.h5` files, the template "fasnacht-hicsdata (HDR)" should be used. As ENVI doesn't handle correctly matrices containing invalid values (NaN), they should be interpolated before using any algorithm.

We also provide hereafter the code to load a `.h5` file in Python and MATLAB. These example are equivalent. We load the data and the wavelengths from the HDF5. The axis of the data should be transposed to be y, x, λ, as commonly used. NaN values in these matrix indicates masked information (due to saturation, bad pixels of the sensor, or mask).

- MATLAB code:

```
data = permute(h5read('file.h5','/hdr'), [2, 1, 3]);
wavelengths = h5read('file.h5','/wavelengths');
imshow(nanmean(data, 3)');
```

- Python code (requires `numpy`, `h5py` and `matplotlib.pyplot`):

```
d = h5py.File('file.h5','r')
data = numpy.transpose(d['/hdr'], (1, 2, 0))
wavelengths = d['/wavelengths']
plt.imshow(numpy.nanmean(data), 2))
```

4.6 Acknowledgements

We would like to thank the Swiss National Cooperative for the Disposal of Radioactive Waste (NAGRA) for participating in the funding of this study, and Jens Becker for his support and advice.

4.7 Author contributions

- LF: drafted the manuscript, developped the computer software, captured the data, processed the hyperspectral data and created the hyperspectral data;
- MLV: drafted the manuscript, selected and characterized the minerals, wrote the text descriptions.
- PB, PR: revised the manuscript

4.8 Competing financial interests

The authors declare no competing financial interests.

4.9 Figures

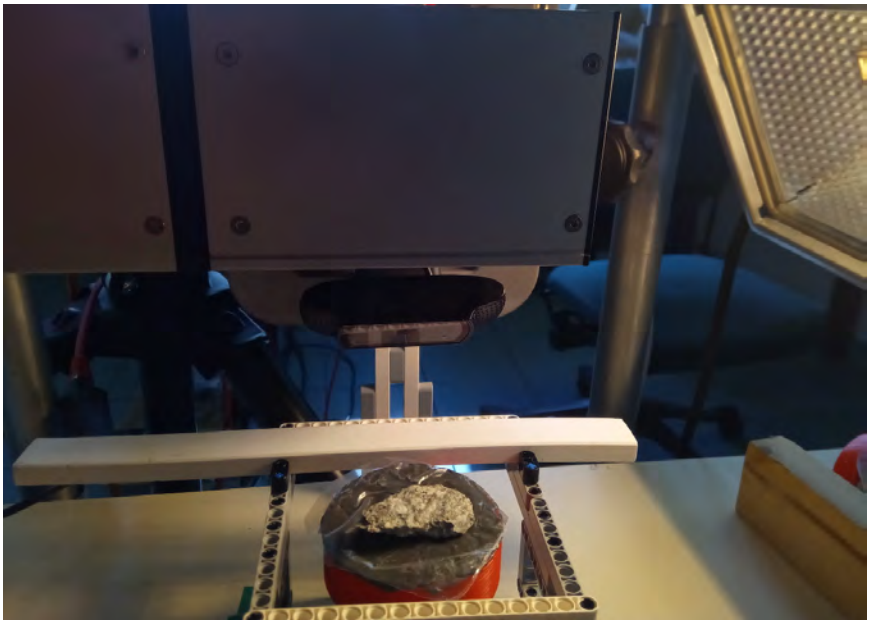


Figure 4.1: Setup to capture data. The sample is put on a support such as its upper surface is approximately horizontal. The hyperspectral camera is above the sample, there is a Logitech C920 webcam behind the sample to capture visible images for photogrammetry. Illumination is done using a halogen lamp (on the right). The white bar is the white reference (Spectralon).

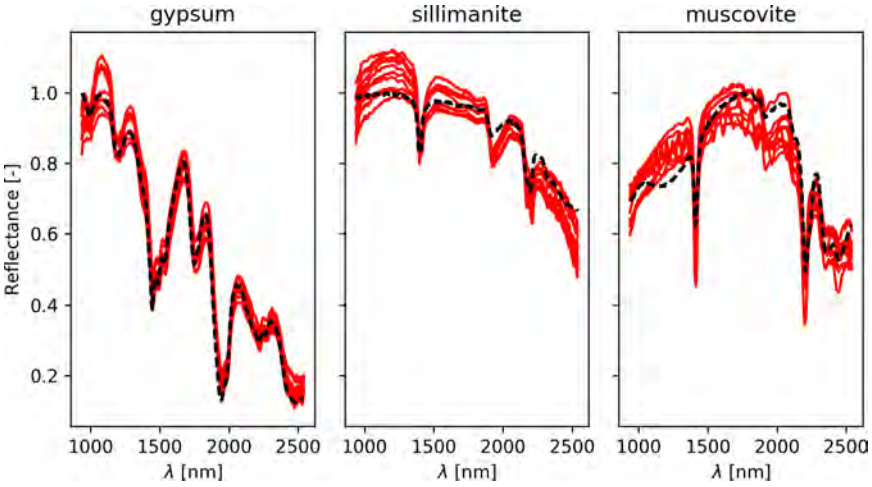


Figure 4.2: Comparison between spectra of the USGS spectral library [65] (in black), and 10 out the 1'000 most similar spectra of the dataset (in red).

4.10 Tables

Table 4.1: Summary of the contents of the dataset

Mineral name	Sample IDs	Datapoints
Actinolite	0020, 0021, 0064	56840
Albite	0107	22521
Almandine	0025, 0026, 0073, 0074	17425
Andalusite	0014	18862
Anhydrite	0004	30832
Apatite	0089(2), 0090(2)	69062
Aragonite	0061	40111
Arsenopyrite	0087(2)	66333
Augite	0038	8503
Barite	0006	37130
Beryl	0075(2)	37743
Biotite	0049(2), 0050	45400
Blende	0086(2)	15596
Bronzite	0112	50452
Bytownite	0103	14666
Calcite	0010, 0011, 0052, 0078, 0079	112501
Cassiterite	0119, 0120	38535
Celestite	0000, 0001, 0002	56075
Chalcedony	0108(2), 0109(2)	96431
Chalcopyrite	0106	19375
Chlorite	0013	75802
Clinochlore	0126	45301
Coal	0081, 0082	44664
Copper	0101	14556
Diopside	0069	58959
Dolomite	0091(2), 0092(2)	76810
Enstatite	0047	31072
Epidote	0023, 0024	47306

Continued on next page

Table 4.1 – continued from previous page

Mineral name	Sample IDs	Datapoints
Fluorite	0003(2), 0012(2)	98638
Galena	0053, 0054	17931
Garnet	0115	27331
Glaucophane	0016, 0017, 0076, 0077	200830
Goethite	0114	10717
Graphite	0083, 0084	20047
Grossular	0030, 0031	46592
Gypsum	0005(2), 0007, 0063	159829
Halite	0008, 0056	66201
Halloysite	0121, 0122	102028
Hematite	0039(2), 0085(2), 0095, 0096	165228
Hornblende	0046	9175
Hypersthene	0111	62932
Ilmenite	0116	16256
Kaolinite	0113	20764
Kyanite	0029	12320
Labradorite	0088(2), 0104(2)	62762
Limonite	0128, 0129	64550
Magnetite	0055, 0072	7103
Microcline	0071	60699
Montmorillonite	0123, 0124, 0125	87986
Muscovite	0034	62481
Nepheline	0097(2)	51930
Olivine	0065	7965
Omphacite	0019, 0067	108032
Opal	0102	34404
Orthoclase	0057	49824
Phlogopite	0045, 0070	105119
Pyrite	0042, 0048	31598
Pyrolusite	0117, 0118	45534
Pyrrhotite	0051	21572
Quartz	0009(2), 0035	126760
Rutile	0093	4959
Sanidine	0099(2)	49260
Serpentine	0018, 0068	78937
Siderite	0080	11651
Silicified wood	0127(2)	92935
Sillimanite	0032, 0033	70546
Sodalite	0043, 0060	61852
Sphalerite	0105	27485
Staurolite	0027, 0028, 0066	30628
Sulfur	0036, 0037	34751
Talc	0022, 0040, 0041, 0058, 0059	138317
Titanite	0094	4121
Tourmaline	0044, 0062	68419
Tremolite	0098(2), 0100	70913
Zircon	0110(2)	5110

Table 4.2: Terms used for mineral classification. Other columns not present in this table are: cleavage, twinning and color.

Classification	System	Habit	Luster	Transparency
native element	monoclinic	euhedral	metallic	transparent
sulfides	triclinic	subhedral	non metallic	translucent
sulfosalts	hexagonal	anhedral	submetallic	non transparent
oxides	orthorombic	cubes	vitreous	opaque
hydroxides	isometric	microcristralline	pearly	
halides	tetragonal	massive	greasy	
carbonates		granular	silky	
nitrates		drusy	adamantine	
borates		lamellar	resinous	
phosphates		foliated	matte	
sulfates		micaceous		
tungstates		tabular		
silicates		bladed		
		fibrous		
		stellated		
		globular		
		botryoidal		
		reniform		
		mammillary		
		colloform		
		dendritic		
		reticulated		
		radiated		
		acicular		
		filiform		
		stalactitic		
		concentric		
		pisolitic		
		oölitic		
		banded		
		amygdaloidal		
		geode		
		concretion		
		prismatic		

4.11 Data Citations

1. Fasnacht, L., Vogt, M., Renard, P. & Brunner, P.. *Zenodo* `http://dx.doi.org/10.5281/zenodo.1446397` (2018)

2. Fasnacht, L., Vogt, M., Renard, P. & Brunner, P.. *Zenodo* `http://dx.doi.org/10.5281/zenodo.1476495` (2018)

3. Fasnacht, L., Vogt, M., Renard, P. & Brunner, P.. *Zenodo* `http://dx.doi.org/10.5281/zenodo.1476503` (2018)

Chapter 5

Reverse engineering hyperspectral cameras, designing efficient software and hardware

The precise hardware and software configuration of a hyperspectral imaging set-up depends on the sample and situation. In particular, there are significant differences between laboratory and field measurements. For example, in the field, the main focus is the simplicity of the installation, as any additional component can create additional problems. In the laboratory, on the other hand, the most important aspect is automation. As a consequence, one may consider, for example, having an automatic focus only in the laboratory, not in the field. This implies that the software used should be very flexible in order to work in many different situations.

As we have seen in chapter 2, adequate software techniques improve the quality of the acquisition. However, to be able to apply these techniques, the control software should be extensible, flexible and easily automatable, which were not the main strengths of the software provided by the manufacturer. The software provided was only able to do back-and-forth scans, had no support for automation (which is required for HDR imaging for instance), was only providing magnitude plots to do focus, and was not able to control any other hardware elements than the ones provided by SPECIM. In addition to these limitation, the software and hardware set-up was very unreliable, and in numerous occurences the controlling laptop had to be restarted.

We therefore had to implement a new control software, named `hics` (Hyperspectral Imaging System Controller), which is specialized in control-

ling push-broom hyperspectral cameras. The main aspects considered when developing this software were:

- the ability to control the hardware, in all the various possible setups.
- the ease of use. (see the user manual in appendix A)
- the simplicity of extending the software. (see section 5.4)

There are similiarities between hyperspectral setups. Usually, a camera has a control interface (to specify parameters such as the integration time and the frame rate), a shutter (to capture dark frames) and a capture interface (to retrieve the captures frames). It is also common to have a device to change the location of the acquisition line with respect to the sample, to get a 2D image. However, the exact way this happens depends on the setup. Since it is not possible to cover all specific use cases, due to the high complexity of the hardware which may be involved in a hyperspectral scanning setup, two specific example cases were considered (they cover most of the hardware available, and most of the set-ups are a subset of these two):

- the acquisition setup used for the acquisition of the spectral library presented in chapter 4, an overview is shown in section 5.1.
- the acquisition setup used to test the AisaKESTREL 10 system, which is the camera intended to be mounted on a UAV, shown in section 5.2.

The presentation of the individual components are split as follows:

- Components specific to the SWIR camera are presented in section 5.1.
- Components specific to the AisaKESTREL 10 system are presented in section 5.2.
- Other components, like inhouse developed components, are presented in section 5.3.

For each component, we give information about the hardware, explain how it was designed or reverse-engineered and give some details about its interface.

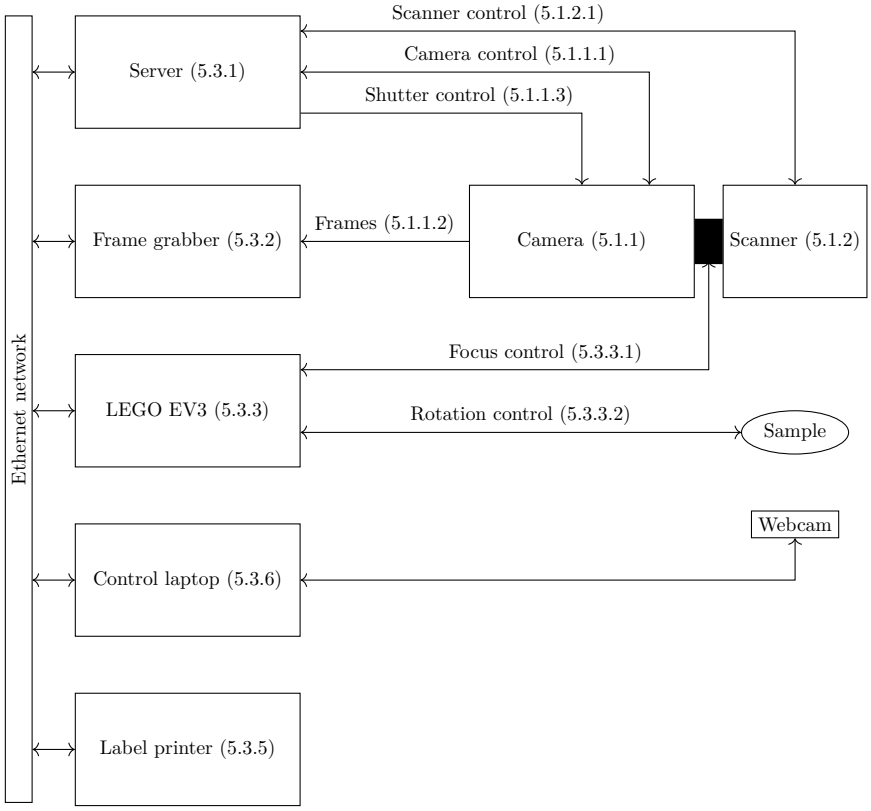


Figure 5.1: Setup for scanning minerals from the university collection. LEGO EV3 is the controller brick of the LEGO Mindstorm system.

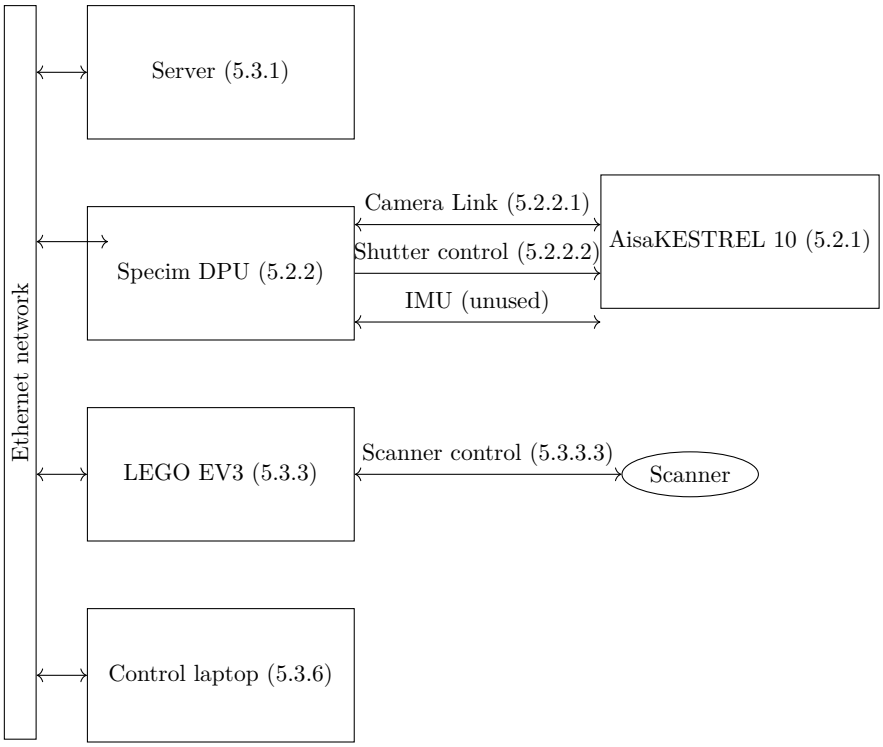


Figure 5.2: Setup when associated with the Specim AisaKESTREL 10 system. Specim DPU (Data processing unit) it provided by Specim to interface the camera. LEGO EV3 is the controller brick of the LEGO Mindstorm system, and in that case controls the orientation of the camera.

5.1 SPECIM SWIR camera and interfaces

The SPECIM SWIR camera is composed of two parts, the camera and a mirror scanner, connected by a proprietary cable. The camera has a data port (unidirectional, frames output) and a power and control port which is connected to the power supply. The power supply exposes the control port as has serial port and a USB interface (on a round connector).

5.1.1 Camera

The camera is a push-broom camera which contains an actively cooled sensor with a resolution of 320×256 pixels. The 320 pixels are the spatial pixels, while the 256 pixels correspond to the bands, which are summarized

in table 5.1. Each pixel is read as a 14 bit value. More details about the sensor characterization can be found in chapter 2.

Table 5.1: SPECIM SWIR camera band centers, in nm.

Bands										
1-10	937.06	943.47	949.88	956.28	962.69	969.10	975.50	981.91	988.31	994.71
11-20	1001.11	1007.51	1013.91	1020.31	1026.71	1033.11	1039.50	1045.90	1052.29	1058.68
21-30	1065.07	1071.46	1077.85	1084.24	1090.63	1097.02	1103.40	1109.79	1116.17	1122.55
31-40	1128.93	1135.31	1141.69	1148.07	1154.45	1160.83	1167.20	1173.58	1179.95	1186.32
41-50	1192.69	1199.07	1205.44	1211.80	1218.17	1224.54	1230.90	1237.27	1243.63	1250.00
51-60	1256.36	1262.72	1269.08	1275.44	1281.80	1288.15	1294.51	1300.86	1307.22	1313.57
61-70	1319.92	1326.27	1332.62	1338.97	1345.32	1351.67	1358.02	1364.36	1370.70	1377.05
71-80	1383.39	1389.73	1396.07	1402.41	1408.75	1415.09	1421.42	1427.76	1434.09	1440.43
81-90	1446.76	1453.09	1459.42	1465.75	1472.08	1478.41	1484.73	1491.06	1497.38	1503.71
91-100	1510.03	1516.35	1522.67	1528.99	1535.31	1541.63	1547.94	1554.26	1560.57	1566.89
101-110	1573.20	1579.51	1585.82	1592.13	1598.44	1604.75	1611.06	1617.36	1623.67	1629.97
111-120	1636.27	1642.57	1648.88	1655.18	1661.47	1667.77	1674.07	1680.37	1686.66	1692.95
121-130	1699.25	1705.54	1711.83	1718.12	1724.41	1730.70	1736.98	1743.27	1749.56	1755.84
131-140	1762.12	1768.41	1774.69	1780.97	1787.25	1793.52	1799.80	1806.08	1812.35	1818.63
141-150	1824.90	1831.17	1837.45	1843.72	1849.98	1856.25	1862.52	1868.79	1875.05	1881.32
151-160	1887.58	1893.84	1900.10	1906.37	1912.62	1918.88	1925.14	1931.40	1937.65	1943.91
161-170	1950.16	1956.41	1962.67	1968.92	1975.17	1981.42	1987.66	1993.91	2000.16	2006.40
171-180	2012.64	2018.89	2025.13	2031.37	2037.61	2043.85	2050.09	2056.32	2062.56	2068.79
181-190	2075.03	2081.26	2087.49	2093.72	2099.95	2106.18	2112.41	2118.64	2124.87	2131.09
191-200	2137.31	2143.54	2149.76	2155.98	2162.20	2168.42	2174.64	2180.86	2187.07	2193.29
201-210	2199.50	2205.72	2211.93	2218.14	2224.35	2230.56	2236.77	2242.97	2249.18	2255.39
211-220	2261.59	2267.79	2274.00	2280.20	2286.40	2292.60	2298.80	2304.99	2311.19	2317.39
221-230	2323.58	2329.78	2335.97	2342.16	2348.35	2354.54	2360.73	2366.92	2373.10	2379.29
231-240	2385.47	2391.66	2397.84	2404.02	2410.20	2416.38	2422.56	2428.74	2434.92	2441.09
241-250	2447.27	2453.44	2459.61	2465.79	2471.96	2478.13	2484.30	2490.46	2496.63	2502.80
251-256	2508.96	2515.13	2521.29	2527.45	2533.61	2539.77				

5.1.1.1 Control interface

The camera control interface is accessible using the serial port of the power supply of the camera. The protocol is not described in the documents that were available, and therefore was reverse engineered using a man-in-the-middle (MITM) approach. In practice, this consisted in having a computer with two serial ports, one connected to the camera and the other one connected to the serial port of the computer running the software provided by SPECIM. The baud rate was found to be 9600. The computer was configured to log and forward the message between the serial ports. By systematically changing all parameters of the camera on the software provided by the manufacturer, it was possible to recover the message format, and a draft of the command syntax. The messages are structured in packets, which contents are described in table 5.2. The checksum was found to be the 8 least significants bits of the sum of the data bytes. The command sent is a comma separated list of the command name, followed by the arguments. The reply is also a comma separated list, which containts the command, then a status code (1 is OK), and then the return value(s) if any. A summary of the commands is given in table 5.3.

During the reverse engineering process, a leaked confidential technical

document of CEDIP infrared systems was found on the internet, for a camera which has similar protocol. It helped confirm that the reversed engineering information was indeed correct.

A Python 3 implementation is provided in [69, hics/hardware/specim-swir/camera.py].

Table 5.2: Camera packet structure

Byte	1	2	3	$4 \ldots 4+n-1$	$4+n$	$5+n$
Content	2	n	0	data (n bytes)	data checksum	3

Table 5.3: Camera commands

Command	Parameter	Reply data	Description
CCM	-	-	Command use to check that the camera is online and works
FRM	Frame rate (frames per second)	-	Set frame rate
GFR	-	Frame rate (frames per second)	Get frame rate
API	-	-	Activate bad pixel interpolation
DPI	-	-	Disable bad pixel interpolation
SEX	(0—1)	-	Activate (1) or disable (0) external triggering
GEX	-	(0—1)	Return 1 if external triggering is enabled, 0 otherwise
DAG	-	-	Unknown, run at camera startup (perhaps "Disable automatic gain")
INT	Integration time in µs, 0, 0, 1 (?)	-	Set integration time
GIT	-	Integration time in µs, 0 (?)	Get integration time
ANU	NUC table ID	-	Activate non-uniformity correction
BNU	-	-	Disable non-uniformity correction
GNU	-	NUC table ID	Get current non-uniformity correction table
GRM	-	0, 0 (?), bitmask	Get remapping (LSB bit: vertical, second bit: horizontal)
FIV	-	-	Toggle vertical flip
FIH	-	-	Toggle horizontal flip

5.1.1.2 Data interface

The data interface of the camera is a SCSI-like 68 pins low voltage differential signaling (LVDS) interface. It is connected to a provided PCI acquisition card, a National Instruments (NI) IMAQ PCI-1422. The signals were not investigated, nor the direct interfacing of the card. For more details about the data acquisition, see section 5.3.2. A C++ implementation for Windows of the frame grabber is provided in [69, helpers/specimswir/NiRawStreamer].

5.1.1.3 Shutter control interface

The shutter interface is a USB port which is connected to a FTDI[1] chip. While FTDI chips are commonly used as USB to serial converters, in the present case it is used in bitbanging mode on 4 bits. This made the classical MITM approach not possible. This interface was reversed engineered by capturing API calls to FTDI D2XX direct driver, which is present as a DLL. The commands are summarized in table 5.4. A C implementation is provided in [69, helpers/specimswir/shutter/shutter.c].

Table 5.4: Shutter commands, each nibble is issued as a independant write command.

Nibbles	Description
`F7F7FCEF`	Initialization
`7FCEF`	Open shutter
`7FAEF`	Close shutter

5.1.2 Scanner

In this setup provided by SPECIM, the scanner is a device containing a mirror, which is mounted between the lens of the camera and the sample. The mirror can be rotated using a motor. Precise movement of the mirror is important to get an image with regular pixel size, therefore the build and the motor have to be of good quality.

5.1.2.1 Scanner control interface

Using a MITM approach, it was possible to determine that the scanner was driven by a Schneider Electric MDrive motor. A MDrive motor is a step-by-step motor with a programmable controller. As the command set and the logic are available [70], it was not required to do command level analysis. However some reverse engineering was still applied to understand the motor parameters and the logic of the application provided by SPECIM. The motor requires 4'096'000 steps to do a full 360 degrees rotation of the mirror. The application programs the controller to perform a back-and-forth movement, and to send a message on the serial port at each change of direction. This was used by the application to split the data at the correct frame.

This approach was dismissed because it is not very practical. Instead, direct controlling of the motor gave very good results, as the code was straightforward, and the latency was low and repeatable (around 20ms). The reason this approach was not used by SPECIM is unknown, but may be related to reliability issue of their application and/or of Windows.

[1] FTDI is a well-known manufacturer of USB to serial converters

A Python 3 implementation is provided in [69, hics/hardware/specim-swir/scanner.py].

Table 5.5: Scanner commands used

Command	Reply data	Description
`EM=2`	-	Disable echo of the commands
`E`	-	Ends the current movement
`MA <pos>`	-	Move the mirror to position
`PR P`	`<pos>`	Get the current position
`PR MV`	`0` or `1`	Returns 1 is the motor is moving
`VM <max_velocity>`	-	Set the maximum velocity (steps per second)
`P VM`	`<max_velocity>`	Get the maximum velocity (steps per second)

5.2 SPECIM AisaKESTREL 10 camera and interfaces

SPECIM AisaKESTREL 10 is a push-broom hyperspectral camera system designed to be light enough to be able to be carried by a UAV. It has the ability to capture visible and near infrared wavelengths. As a proof of concept, an interface was developed to connect the DPU to the acquisition software developed for the SPECIM SWIR camera.

This development required only 3 days (mostly spent doing reverse engineering). The resulting Python 3 implementation, intended to be run on the Data Processing Unit (DPU), is provided in [69, hardware/specimAisaKestrel10/camera.py].

5.2.1 AisaKESTREL 10 camera assembly

The camera assembly consists of an inertial measurement unit (not used by the software presented here), a Photon Focus camera and an assembly from SPECIM. The camera is connected directly to the DPU using a CameraLink interface. The assembly is connected directly to the DPU using a proprietary cable, and exposes a serial port.

The assembly contains the shutter (see section 5.2.2.2). It also contains the optical components required to capture a 1D line of light, and to create a 2D image on the sensor of the Photon Focus camera. The image captured has a dimension of 2048×2048 pixels, each having a depth of 12 bits. In practice, only a part of the sensor is used, of size 2048×356 pixels. The first dimension is spatial, while the second corresponds to the bands (see table 5.6).

Table 5.6: SPECIM AisaKESTREL 10 camera band centers, in nm.

Bands										
1-10	394.7	396.39	398.07	399.76	401.44	403.13	404.82	406.5	408.19	409.88
11-20	411.57	413.25	414.94	416.63	418.32	420.01	421.7	423.39	425.08	426.77
21-30	428.47	430.16	431.85	433.54	435.24	436.93	438.62	440.32	442.01	443.71
31-40	445.4	447.1	448.79	450.49	452.18	453.88	455.58	457.28	458.97	460.67
41-50	462.37	464.07	465.77	467.47	469.16	470.86	472.56	474.26	475.97	477.67
51-60	479.37	481.07	482.77	484.47	486.18	487.88	489.58	491.28	492.99	494.69
61-70	496.4	498.1	499.81	501.51	503.22	504.92	506.63	508.33	510.04	511.74
71-80	513.45	515.16	516.87	518.57	520.28	521.99	523.7	525.41	527.11	528.82
81-90	530.53	532.24	533.95	535.66	537.37	539.08	540.79	542.5	544.22	545.93
91-100	547.64	549.35	551.06	552.77	554.49	556.2	557.91	559.63	561.34	563.05
101-110	564.77	566.48	568.19	569.91	571.62	573.34	575.05	576.77	578.48	580.2
111-120	581.91	583.63	585.34	587.06	588.78	590.49	592.21	593.93	595.64	597.36
121-130	599.08	600.8	602.51	604.23	605.95	607.67	609.39	611.1	612.82	614.54
131-140	616.26	617.98	619.7	621.42	623.14	624.86	626.58	628.3	630.02	631.74
141-150	633.46	635.18	636.9	638.62	640.34	642.06	643.78	645.5	647.22	648.94
151-160	650.67	652.39	654.11	655.83	657.55	659.28	661.0	662.72	664.44	666.16
161-170	667.89	669.61	671.33	673.05	674.78	676.5	678.22	679.95	681.67	683.39
171-180	685.12	686.84	688.56	690.29	692.01	693.73	695.46	697.18	698.91	700.63
181-190	702.35	704.08	705.8	707.53	709.25	710.97	712.7	714.42	716.15	717.87
191-200	719.6	721.32	723.04	724.77	726.49	728.22	729.94	731.67	733.39	735.12
201-210	736.84	738.57	740.29	742.02	743.74	745.47	747.19	748.92	750.64	752.36
211-220	754.09	755.81	757.54	759.26	760.99	762.71	764.44	766.16	767.89	769.61
221-230	771.34	773.06	774.79	776.51	778.24	779.96	781.69	783.41	785.13	786.86
231-240	788.58	790.31	792.03	793.76	795.48	797.2	798.93	800.65	802.38	804.1
241-250	805.83	807.55	809.27	811.0	812.72	814.44	816.17	817.89	819.62	821.34
251-260	823.06	824.79	826.51	828.23	829.95	831.68	833.4	835.12	836.85	838.57
261-270	840.29	842.01	843.74	845.46	847.18	848.9	850.62	852.35	854.07	855.79
271-280	857.51	859.23	860.95	862.68	864.4	866.12	867.84	869.56	871.28	873.0
281-290	874.72	876.44	878.16	879.88	881.6	883.32	885.04	886.76	888.48	890.2
291-300	891.92	893.64	895.35	897.07	898.79	900.51	902.23	903.95	905.66	907.38
301-310	909.1	910.82	912.53	914.25	915.97	917.68	919.4	921.12	922.83	924.55
311-320	926.26	927.98	929.69	931.41	933.12	934.84	936.55	938.27	939.98	941.7
321-330	943.41	945.12	946.84	948.55	950.26	951.98	953.69	955.4	957.11	958.82
331-340	960.54	962.25	963.96	965.67	967.38	969.09	970.8	972.51	974.22	975.93
341-350	977.64	979.35	981.06	982.77	984.48	986.18	987.89	989.6	991.31	993.01
351-356	994.72	996.43	998.13	999.84	1001.55	1003.25				

5.2.2 Data processing using (DPU)

The DPU is a box provided by SPECIM which controls the camera. It contains the power supply, a Windows 7 computer (Intel NUC), and a EPIX PIXCI mini PCI-Express frame grabber. By default, it starts an application which allows data capture and recording, and remote control.

The DPU has a Gigabit network interface configured as 192.168.168.2/24, which is normally used for the radio modem, to connect to the ground station. This interface has been used to connect to the server (which was configured to have one additional address of 192.168.168.1), in order to capture the frames in real time.

5.2.2.1 Camera Link

The Camera Link interface of the camera is connected to the EPIX capture card in the DPU. It serves both to control the camera (serial port) and to capture the frame data. The manufacturer of the capture card (EPIX) provides a library and a software development kit (SDK) to use both functionalities [71]. The library is used by SPECIM, and is therefore present on the DPU. Unfortunately the SDK is not freely available, therefore a work-around had to be found. Enough information was found on internet to guess the prototypes of the API calls required to capture data and to use to serial port. Using Python `ctypes.windll`, it was possible to directly call the library.

The commands of the camera are described in the Register based ASCII Protocol, provided by Photon Focus [72]. A defensive implementation was made in [69, hardware/specimAisaKestrel10/camera.py], which restricts the available register available in order to avoid mis-configuring the camera by mistake.

5.2.2.2 Shutter control

The shutter control is exposed on a serial port, which is configured at a speed of 115'200 bauds. Due to the proprietary connector used, it was not possible to use a MITM approach. Therefore, the API calls were intercepted instead, in order to capture the commands to open and close the shutter (table 5.7). Due to the length and the complexity of the commands, it seems likely that other commands are implemented, but that was not investigated as it was not required to use the camera.

Table 5.7: Shutter commands.

Bytes	Description
`A5,D2,77,12,00,00,00,02,10`	Open shutter
`A5,D2,77,12,00,00,00,01,13`	Close shutter

5.3 Custom designed components

The components mentioned in the previous sections are provided by the camera manufacturer. However, to improve the usability of the setups, a number of custom components were developed. These components are described in the following subsections.

It should also be noted that the cabling and carriability of the setup is important. Therefore, the camera power supply, the frame grabber and the server were mounted in an audio 19 inch rack, which is designed to be carried. This has been shown to be very valuable for field work (fig. 5.3).

Figure 5.3: Example of working conditions in the Grimsel underground laboratory. The control laptop (which has the GUI) is laid on top of the box containing the camera power supply, the server and the frame grabber.

5.3.1 Server

The server is the main component of the infrasture. It organizes the network communication and serves as a host computer to interface the SWIR camera. Additionally, it also does some of the processing (dark frame subtraction for the preview, autofocus algorithm, etc.).

For the network communication, it runs:

- the DHCP[2] server for internal network, in order to allocate network settings for all the devices connected to the internal network.
- the iSCSI[3] server to provide storage for the frame grabber (more details in section 5.3.2)
- the Redis server, which by the various software components to communicate

[2]DHCP is a protocol used for network device autoconfiguration

[3]iSCSI is a technology which allows block device access (e.g. hard drives) through a computer network

- the gateway for internet access (which is used only in the laboratory)

5.3.2 Frame grabber

As the NI IMAQ card has only Windows driver, it was required to have a Windows computer to capture the data. However, since Windows is difficult to run reliably without interaction for a long time, a few "tricks" were used:

- A minimal software was developed in order to capture and stream directly to the network, `NiRawStreamer` [69, helpers/specimswir/NiRawStreamer].
- Linux on the server was configured to create a copy-on-write[4] (COW) device, which is exported as iSCSI, which is used as the boot drive on the frame grabber. This has the advantage that after each reboot of the server, the frame grabber computer will start at exactly the same state.
- The Windows 7 image used to boot is minimal.
- The DHCP server doesn't give a default route to the frame grabber, which prevents Windows to get internet access (and therefore also prevents updates).

This setup requires a correct boot sequence of the various computers, as otherwise the frame grabber won't have access to the network configuration and ressources needed. The boot process is as follows:

- The system is configured for network boot, and to stay off when it gets power. Of course, its BMC starts.
- The server starts and exports the COW disk image as a iSCSI hard drive.
- When the server is fully ready, it triggers via IPMI the startup of the frame grabber.
- The frame grabber boots via PXE[5], downloads iPXE[6] via TFTP[7], which configures the iSCSI boot, and then starts Windows.
- Windows finally launches `NiRawStreamer`.

The setup to create the overlay being uncommon, the gist of its configuration is given hereafter: the commands (listing 5.1), the config file for the iSCSI Entreprise Target Daemon (listing 5.2), the excerpt of the DHCP configuration (listing 5.3) and the iPXE boot script (listing 5.4).

[4] Copy-on-write means that the data written is written not on the original file, but instead somewhere else

[5] PXE (Preboot eXecution Environment is a system which enables booting a device from network

[6] iPXE is a tool which extend the functionality of PXE

[7] TFTP (Trivial file transfer protocol) is a very simple way to transfer files on the network. It is used by PXE because of its simplicity

Listing 5.1: Commands to create the COW disk image

```
/sbin/modprobe zram num_devices=1
/bin/bash -c 'echo 268435456 > /sys/block/zram0/disksize'
/sbin/losetup /dev/loop0 /srv/win7.img
/sbin/losetup /dev/loop1 /dev/zram0
/sbin/dmsetup create win7-sandbox --table "0␣41943040␣snapshot␣
    ↪ /dev/loop0␣/dev/loop1␣N␣1"
```

Listing 5.2: /etc/ietd/ietd.conf

```
Target iqn.2014-10.ch.unine.chyn.hsc:san.win7-1
        #Use this line instead of the next one to use directly
            ↪ the disk image
        #Lun 0 Path=/srv/win7.img,Type=fileio
        Lun 0 Path=/dev/mapper/win7-sandbox,Type=fileio
```

Listing 5.3: Host configuration in /etc/dhcpd/dhcpd.conf

```
host capture {
  hardware ethernet 00:25:90:ca:92:f6;
  fixed-address 192.168.1.128;
  if exists user-class and option user-class = "iPXE" {
    filename "/boot.ipxe";
  } else {
    filename "intelx.kpxe";
  }
}
```

Listing 5.4: /srv/tftp/boot.ipxe

```
#!ipxe

set san-rootfs iscsi:192.168.1.2:::0:iqn.2014-10.ch.unine.chyn.
    ↪ hsc:san.win7-1
set keep-san 1

#Comment this line to only mount the iSCSI target but not to
    ↪ boot from it
#This is useful to install Windows on the image for example.
sanboot --drive 0x80 ${san-rootfs}

sanhook --drive 0x81 ${san-rootfs}
exit
```

5.3.3 LEGO Mindstorm EV3

The LEGO Mindstorm EV3 is often considered to be a toy, but it is in reality an inexpensive and versatile robotic platform. The main part is the control brick, with 4 sensors input and 4 motor output. To implement the logic, LEGO proposes a graphical programming language. While it is great to learn programming, it is not convenient for the needs of a hyperspectral

scanning. Hopefully, it is possible to get a more standard environment with the help of ev3dev [73]. ev3dev is a Debian-based Linux distribution which can run on the control brick, and provides also Python 3 modules to interact with the LEGO hardware. With the addition of a USB-to-Ethernet adapter, it is possible to use the LEGO control brick exactly like a "normal" computer, except that it has access to the LEGO sensors and actuators.

We mostly used only the motors. The motors however have very advanced features, that are normally only available on expensive devices: they have a angular encoder, so their position can be queried at all time. This means that is is possible to control them by setting the target position. Moreover, their speed can be controlled either using the duty cycle, or by giving a target angular speed and a maximum acceleration. Using gears, it is possible to create very precise movements. The main limitations are the play between gears, and the elasticity because everything is made of the plastic.

Each of the following constructions uses very similar control code, as their basic function is to turn a motor, so the details will only be given about the mechanical setup.

5.3.3.1 Focus control

While focus can be done manually, or in an operator assisted way (see section 2.4), automating it is important for scanning efficiency, for example for the acquisition of data presented in chapter 4. Of course, this requires mechanical control of the focus rig. The first build was using a belt to turn the focus rig, but it was not satisfactory, since there were some elasticity in the control and the belt had a tendancy to slip. Also, it was exercing a constant lateral force on the lens, which creates mechanical strain on the camera.

The second build is the one that is currently used, and has shown to be satisfactory. It uses a 3D printed adapter (fig. 5.4) which is mounted on the lens. It provides a LEGO compatible gear and holds the camera lens very firmly. As a safety measure, its mechanical design prevents the unscrewing of the lens (which would make it fall), even in case of a software bug. This also requires a 3D printed plate, to hold a LEGO gear at exactly the right distance of the focus rig (fig. 5.5).

The complete assembled element in shown in figure 5.6, and the code is available in [69, hics/hardware/ev3/ev3focus.py]. An example algorithm to auto-focus is available in [69, hics/plugin/autofocus.py].

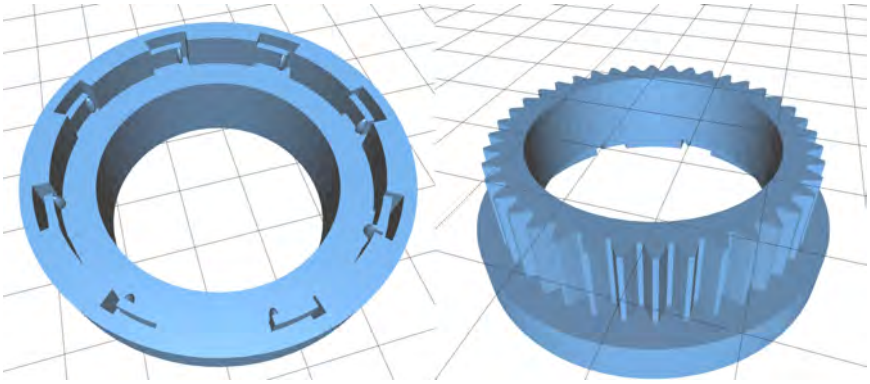


Figure 5.4: Adapter to allow lego gear to act on the focus rig, designed to be 3D printed.

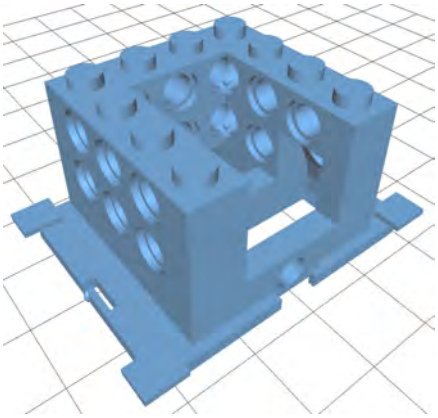


Figure 5.5: 3D printed part to have the lego gear at the right distance of the adapter.

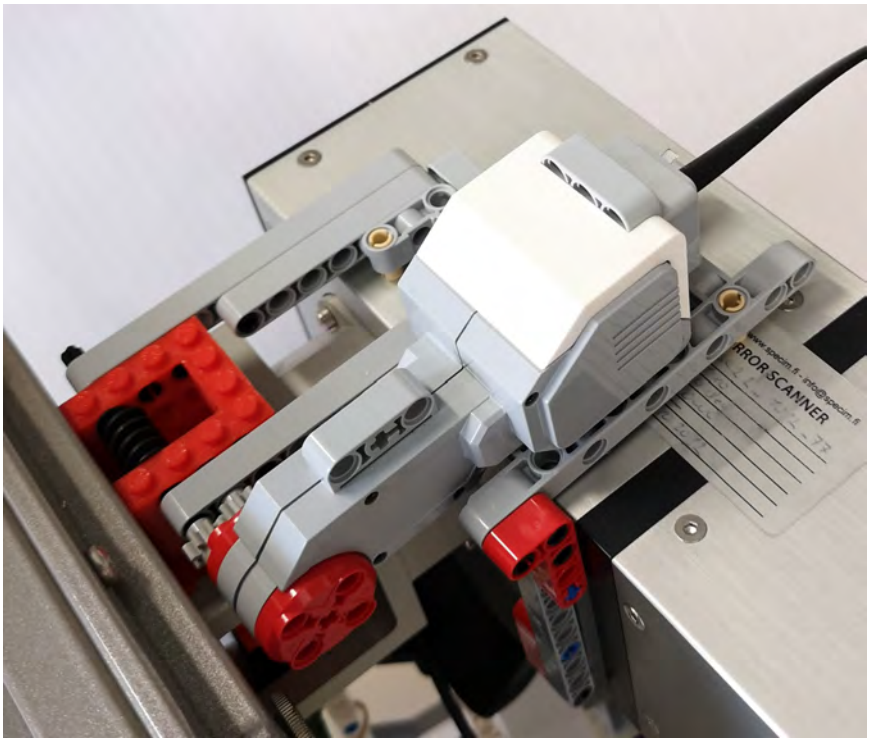


Figure 5.6: Fully assembled focus control.

5.3.3.2 Rotation control

Another useful feature is to be able to rotate the sample on the horizontal plane. It enables performing photogrammetry (using the webcam) or taking multiple images of the sample at different illumination.

It consists in two different parts:

1. A 3D-printed sample holder, which exists in two versions. One is designed to be filled with plastilin modeling clay for scanning rock samples (fig. 5.8). The other is a "spoon-like" shape for holding powders. (fig. 5.9)

2. A LEGO assembly (fig. 5.7).

The code is available in [69, hics/hardware/ev3/ev3rotation.py].

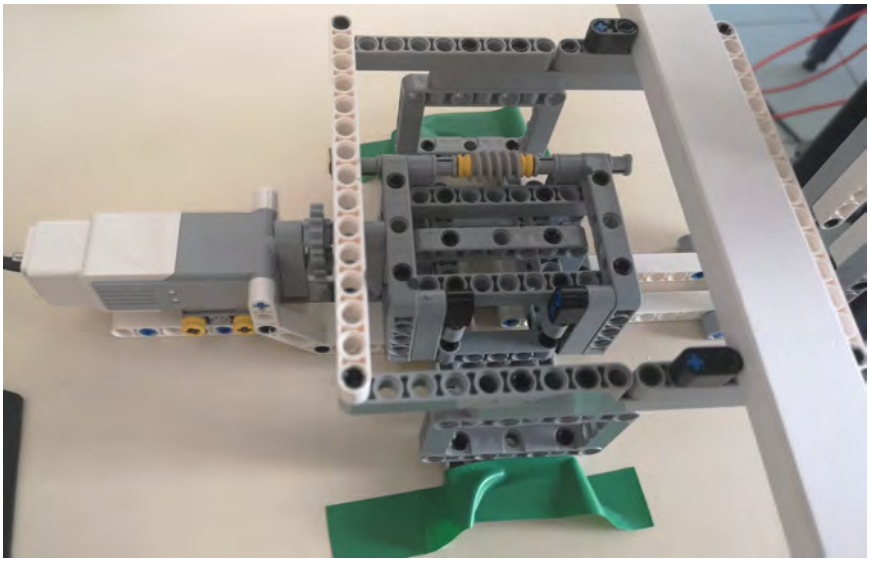


Figure 5.7: LEGO assembly for the rotation. A white spectralon is fixed along, to enable easy calibration.

Figure 5.8: 3D printed part for rotating a mineral sample.

Figure 5.9: 3D printed part for rotating powders.

5.3.3.3 Scanner

The AisaKESTREL 10 camera is intended to be mounted on a UAV, therefore it has no builtin scanner. To get a 2D image, a very simple 3D printed device was designed (fig. 5.10).

The code is available in [69, hics/hardware/ev3/ev3scanner.py].

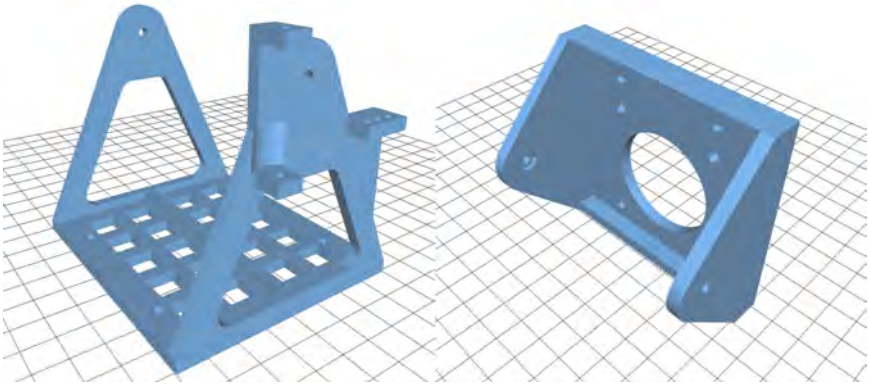


Figure 5.10: 3D printed parts to hold and rotate the AisaKESTREL 10 camera.

5.3.4 Scanner using the macro objective

Using a macro objective, it is possible to scan samples at very high resolution (the pixel size is around 30 µm). However, the mirror scanner of the camera is not steady enough to image at this precision, and it is therefore required to move the sample. It is obviously impossible to be precise enough using

LEGOs, and commercial motorized linear stage are expensive (around 5000CHF).

Therefore, a precision stage for a drill was bought (around 70CHF), and a Thorlabs motor was adapted to be mounted on the X axis (fig. 5.11).

Using this setup, it was possible to get a precision of about 10 µm on the motorized axis.

On the software side, a software library to communicate with the motors was written [74]. The specific code to control the motor as a scanner is available at [69, hics/hardware/thorlabs/scanner.py].

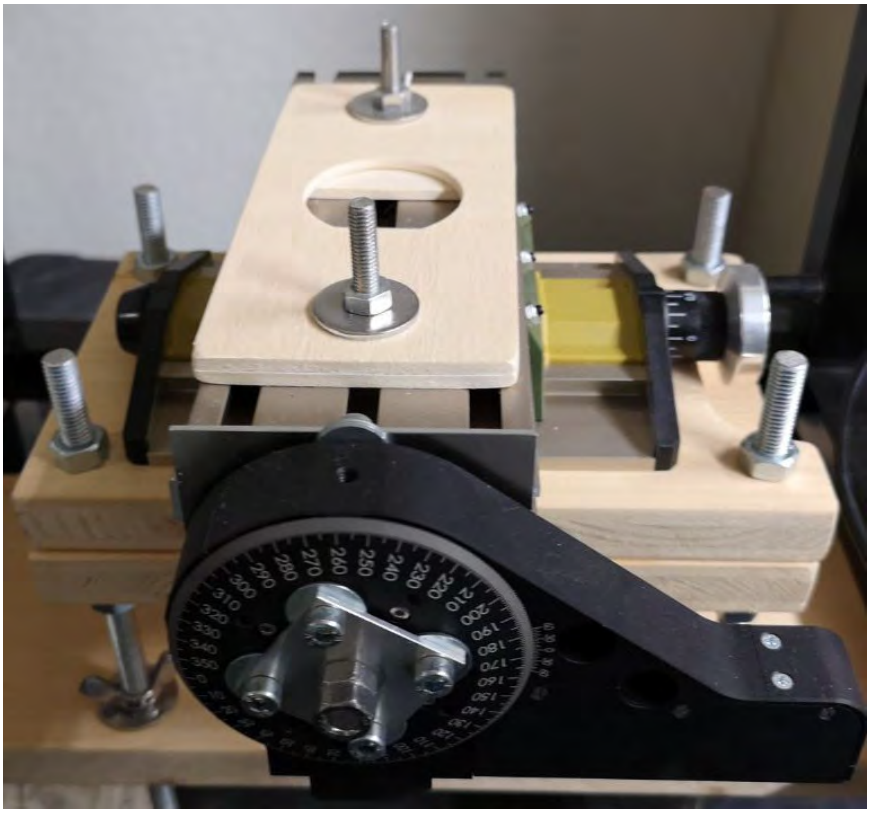


Figure 5.11: Precision stage using Thorlabs motor.

5.3.5 Label printer

A Brother QL-820NVB label writer was added to the system, in order to automate the printing of the labels of the sample scanned. The general printing language of the printer is complex, but there is a template system.

The template system allows a label to be prepared using Brother's software, and then to have fields "replaced" by any value when printing. This has the advantage of limiting the simple code which only provides these values.

A plugin was written in order to take advantage of this procedure. Although the file is 50 lines long, only 10 lines are required to print. The code is available in [69, hics/plugin/labelprint.py].

5.3.6 Control laptop

The control laptop is running the graphical user interface (GUI), to allow the operator to control the whole system. It also controls the webcam, as it is usually physically near the sample (the operator needs to have access both to the sample and the computer).

In general, it also records the data from the camera, as it is efficient to have the data on a computer which has a screen.

More details about the control software are available in the user manual, in appendix A.

5.4 Software design

Throughout this section, only the general concepts of the software design are covered. Specific details about the implementation should be looked up directly in the code.

There are two key requirements for this kind of software: reliability and flexibility. Reliability is important during the data acquisition, because one has to be efficient, especially since the data may be only captured once. During the data processing, reliability is related to the reproducibility, it is important to know what processes exactly have been applied to the measured data.

Flexibility, one the other hand, is needed because the setup can vary significantly. For this reason, the software is constructed in small components which communicate between each other. A key aspect of the software design is the plugin interface. It allows easy development of extensions which have simultaneoulsy a control logic (like changing the parameters of the camera) and capture data. The interface also allows for asking user input, and providing live output (plot).

To allow easy communication between components, it was decided to base everything on Redis [75]. Redis is conceptually based on keys, which are an arbitrary string of characters. Keys can be used in two different ways: as the key of a key-value store (KVS) or as the name of "topic" (channel) in a Publish-Subscribe (PubSub) mechanism. A key value store is basically a dictionary, and allows storage of arbitrary data for each key. The value can be retrieved by any connected programs by reading this key. An additional

feature of the KVS is expiration, which makes the entry self-destruct after a given time. The PubSub mechanism allows message passing between programs. A program can send a message to a given channel, and every connected program (if any) which were subscribed to this channel at that moment will receive the message.

The programs developed extensively use these mechanisms. Each property (like the position of the scanner, the integration time, etc.) is stored in the KVS, while events (property change, frames, etc.) are broadcasted using the PubSub mechanism. This approach creates a highly decentralized and flexible environment, which has nice properties, such as the possibility of simultaneous use of the same data. For example, a plugin can record the data, while it is simultaneously displayed on two different computers running the GUI. The only constraint is that only one component is allowed the control at a given time. Additionally, each component is only defined by its interface, which allows easy substitution by a different implementation. This is very valuable when using different hardware which have the same functionality in the system (for example, when replacing the mirror scanner by a scanner which instead moves the object).

Data storage is done using mmapickle files [76], because it allows working with quantity of data larger than the size of the memory of the computer and because it allows looking at the file contents while it is written.

5.4.1 Data acquisition

Data acquisition is a critical phase, as it requires to be efficient, fast, and robust. Moreover, it has to be simple to use, as the working conditions can be difficult when in the field (fig. 5.3).

To restrict to one component using the camera at a time, a lock implemented in redis is used. The trick is to use a specific key with an expiration of 10 seconds. When a component wants the lock, it has to create this key and set the value to its own name and keep refreshing it. If it fails to refresh it, then the lock has expired and the component is not allowed to control the camera any more. If the component wants to launch another component which requires the lock, it can delete the key (release the lock) just before launching the other component, and recreate the key (acquire the lock) as soon as the component has finished. The behaviour of the components regarding the camera control and the lock is a “gentlemen’s agreement” as it is not enforced, and the setup is considered to be running in a trusted environment.

The main component is the GUI, which is written using PyQt5. By design, its functionnality is limited to:

- providing an overview of the current settings of the various and the ability to control it

- showing the current frame which is output by the sensor
- showing a history of the previous frames (waterfall plot)
- allowing the launch of plugins (possibily releasing the lock if needed), to ask the user for the plugin parameters and to display the plugin output (plot).
- enabling the user to export the current frame or the contents of a plot.

The main purpose of the GUI is to allow the operator to set up everything correctly and to launch the plugin that will do the automated work. This approach was selected due to the large variety of different data use (high dynamic range, timelapse, direct computation of focus, auto-exposure, etc.).

Plugins are small Python scripts, which run as independant programs and connect to Redis. They create keys which allow the GUI to identify their interface, and wait for a PubSub message to execute their code. Plugins write to Redis notably:

- Their internal name (basis for the keys name)
- Their name in the menu
- What parameters to ask the user when launched
- Whether they require a lock (plugins that control the camera) or not (passive plugins, which only display some additional information)
- What output should be displayed on screen.

The GUI can run in two different modes: read-only mode (which doesn't require the lock) or normal mode (requires the lock). The advantage is that it can be launched as a monitor process (multiple GUIs on different computers for example) and also allows visualization of the current status when plugins are running. By default, it tries to acquire a lock. The lock can however be released manually by the user. When the lock is held, the GUI can launch plugins which require the lock. Otherwise, it can only launch passive plugins.

The data flow is kept short to lessen the risk of problems, and is normally:

1. Raw frames are captured by the frame grabber (published to Redis)
2. The frame converter does the dark frame subtraction if possible, to get trivally corrected frames (published to Redis).
3. The GUI listens and displays the corrected frames.
4. The plugins generally capture raw frames.

5.4.2 Data processing

Once the data file is captured, the process is less sensitive to performance issues, since it can be done after the data acquisition is done. However, it is important to be very transparent on the content of the files and what has been applied to them. For this reason, every file can be displayed using the viewer program (`hics.viewer`), and also the contents of a file can be listed using a command line utility (`hics.datafile.ls`).

Each file is a mmappickle dictionary which contains at least a plain text description of the content of the file and a list of wavelengths. It may contain any additional data, like the various captures images, dark frames, white frames, and any additional metadata.

To enable the user to have the flexibility to choose what processing he wants to apply, every processing step is an individual program. While they can vary, usually one would like to do the following (this can be easily automated using a `Makefile`, like the one shown in listing 5.5):

1. Capture the data
2. Compute the reflectance, masking the bad pixels (`hics.datafile.calibrate`)
3. Merge the images, high dynamic range (`hics.datafile.merge`)
4. Create a mask
5. Apply the mask on a file (`hics.datafile.maskdata`)

Of course, processing steps may have parameters. To ensure reproducibility, the files are not modified. Instead, a new one is created, containing the results of the operation, the processing steps of the input files, and the current processing step details. Tables 5.8 shows an example of such a table. The advantage of that table is that it carries all the information required to reproduce the result (the exact code version and the parameters). It also enables the researcher to recompute every image that would be affected by a bug in an algorithm for instance.

Table 5.8: Example processing steps as stored in a file. The list contains the module called, the git revision, and the parameters of the operation. A git revision ending in `-dirty` indicates that some uncommitted changes were made. This is bad because exact code modified in not stored in the file, and therefore the file doesn't contain all the required information.

hics.datafile.calibrate	c876c4ae7023fdb7a17d243c7fbc88c712ed5257
{'max': 16000, 'bad_deviation': 0.1, 'cropframes': None, 'white_variance_threshold': 0.1, 'output': '0070/A.calibrated', 'calibration': None, 'input': '0070/A.scan', 'min': 0}	
hics.datafile.merge	c876c4ae7023fdb7a17d243c7fbc88c712ed5257
{'output': '0070/A.hdr', 'clean': True, 'input': '0070/A.calibrated'}	
hics.datafile.maskdataauto	faedb5ed025465cb1118f8ed8e342d6cc0d52720-dirty
{'output': '0070/A.mhdr', 'maskspec': ['147,209,0.59'], 'input': '0070/A.hdr'}	

Listing 5.5: Example Makefile to process captured data

```
.PHONY: distclean

%.calibrated: %.scan
        python3 -m hics.datafile.calibrate --input $< --output
            ↪ $@ --max 14000

%.hdr: %.calibrated
        python3 -m hics.datafile.merge --input $< --output $@
            ↪ --clean

%.nhdr: %.hdr
        python3 -m hics.datafile.normalize --input $< --output
            ↪ $@

%.png: %.hdr
        python3 -m hics.datafile.maskdata --input $< --mask $@

%.nmhdr: %.nhdr %.mask.png
        python3 -m hics.datafile.maskdata --input $(word 1,$^)
            ↪ --mask $(word 2,$^) --output $@

%.envi.hdr: %.nmhdr
        python3 -m hscc.datafile.asenvi --input $< --output $@

distclean:
        rm -f *.refl *.hdr *.nhdr *.nmhdr *.png
```

Chapter 6

Conclusion and perspectives

6.1 Hyperspectral imaging and mineral identification

Hyperspectral imaging is a promising technology for mineral identification and mapping. However a range of issues currently undermine the robustness of the approach. The whole process from data acquisition to classification was therefore analyzed and improved. The camera was studied and reverse engineered in order to understand exactly its operating principles. Each aspect of the scene set-up, such as the type of light source, the positioning of the various elements of the set-up and the impact of surrounding objects was carefully evaluated. Algorithms were designed to improve the efficiency of the acquisition, such as automate focus or to avoid difficulties in choosing the correct exposition. Methods to combine multiple images in order to increase the data quality were also developed: high dynamic range was implemented to merge multiple images captured with different integration time, and a method was developed to separate the transmittance and the reflectance of translucent objects. The efficiency of artificial neural networks was evaluated for classification. It was found that they were producing highly accurate results, but required complete spectra as input, which requires interpolation of bad pixels and prevents the re-use of an already trained network with a different camera. Therefore a novel input layer was designed which handles efficiently spectra with missing bands. As no training data was readily available for training an artificial neural network for mineral identification, a spectral library of 2D reflectance images of 130 samples of 76 distinct minerals was created. To the best of our knowledge, no similar

dataset exists. Finally, hardware and software tools were designed and implemented to carry out all these steps and to allow efficient automation of all the devices involved.

The study show the importance of correct and documented acquisition procedure in hyperspectral imaging. In particular, illumination is key. It is not only required to have a good light source, but also to avoid backscattering on surrounding objects. Moreover, correct strategies for sensor calibration and reflectance computation are important to avoid misinterpreting the scan results. High dynamic range imaging is also a very valuable technique, as most of the acquision performed involved large variations of intensity. An artificial neural network based on the approach developed is able to reach more than 98% of correct classifications when used with the full library of 76 minerals, which is better than approaches not based on machine learning. The input layer developed was also shown to be very resilient to missing data, as it matches using spectral features instead of the global shape of the spectra.

The procedure presented still has limitation, despite its high overall accuracy. It doesn't detect and perform well on highly reflective objects, such as metallic luster minerals. There is also no indication that this is the best approach possible, and it is possible that other types of wavelets, or other transforms (such as convoluting the wavelet with the spectrum) yield to better results. Another is that we considered only spectral information, and did not use any spatial information.

Indeed, artificial neural network classifiers can be explored beyond what has been done in this thesis. Other types of transforms should be investigated. Techniques combining spectral and spatial information should also be evaluated, for example by using the proposed input layer on a convolutional neural network. For this purpose, the spectral library should be extended to also contain rock samples. It is probably possible to use a semi-supervised approach to create such extension, by using the already existing dataset as a base classification for the new samples.

Experiments have also been made using neural network to predict scalar values, such as surface water content. It is hard to get enough data, but it seems possible to combine quantitative (specific water content) and qualitative data (the sample is drying) in order to train a neural network. The key trick is to use the same neural network multiple times simultaneoulsy, and use a comparison function in order to constrain the function predicting the surface water content to have some properties, like monotony. These experiments should definitely be persued.

As this approach yield very satisfactory results, possible applications should be investigated. The Swiss National Cooperative for the Disposal of Radioactive Waste (NAGRA) is interested in using hyperspectral imaging to determine clay composition. Unmixing approaches were not accurate

enough, but it seems likely that the approach presented in this thesis could be used to create a hyperspectral library of clays of different compositions, and to use an artificial neural network to classify measurements.

6.2 Personal thoughts on engineering aspects

I'd like to conclude this thesis by providing a personal opinion on some engineering aspects related to the work performed. The key difficulty with the tools provided with the camera is the lack of control over the acquisition process. For instance, when performing data acquisition with the software provided by the manufacturer, it is generally impossible to automate the control of the camera, or to get direct access to the data measured. This is inadequate for research. Without automation, one has to control the camera manually, which makes any algorithm improving the acquisition tedious to apply. Moreover, there are often only scarce information about the algorithms applied automatically on the data. There are also similar problems during the data processing steps: often, the processing is done using proprietary software, in which the user has little control over the exact algorithm applied. In this situation, the user of the acquisition system has to trust the correctness of the implementation, and has only very limited ways to try new ideas.

A very simple example can be used to illustrate the shortcomings of the present situation: bad pixels. Often, data produced by cameras hide them, by interpolation which happens either in software or in hardware. This interpolation has visible consequences on the quality of images, and can lead to biaised interpretation if the interpolation is not satisfactory. On the other hand, data gaps can be handled by robust algorithms. Unfortunately, the algorithms implemented in software usually used are not only not robust, but none are designed to even tolerate missing data. They instead return wrong values, stop functioning correctly, or end in an infinite loop. Therefore, the hyperspectral camera manufacturer will always provide interpolated images, which make no demand for software developers to handle such data.

I suppose that these engineering issures are favored by both the manufacturers and the users of hyperspectral acquisition devices: first, there are some incitatives to hardware manufacturers not to provide full information. Providing source code or full specifications about the hardware may show some deficiencies that would have been difficult to spot otherwise. It also has a non-negligible cost to create the documentation and associated support. Reseachers working in earth-science related fields may also focus on the scientific results resulting from measurements, instead of the technical aspects of obtaining these measurements, and therefore not require full technical information from the manufacturer. It is also likely that physics, electronic and software engineering is not their field of expertise. This

situation is also probably favored by the fact that software production is not always considered to be an academic achievement.

To allow progress with respect to these issues, it was decided that all software written during this thesis will be published as open source software. I sincerly hope that this will help other researchers to improve their processes and as a consequence will enable them to try different approaches other than the standard ones. This may lead to very interesting scientific results.

There are however also difficulties with the open source approach, the main being how to promote the software and how to encourage people to use it. There is also the question about how to ensure perdurable support for users of this software, as this has shown to be required for other tools. For promotion, I think that having a complete documentation available is usually enough, as people tend to use search engines. For providing support, the main issue is funding. It may be possible to sell services (like billing people to implement the software required to support their camera).

Appendices

Appendix A

Hyperspectral Imaging Controller Software User Manual

A.1 Introduction

In this chapter, we will present how to set up and use a minimal hyperspectral imaging controller software (hics) system. By hics system, we consider all the components which interact to enable hyperspectral data acquisition.

As seen in chapter 5, the basic system requirement is Redis, which should be accessible by any components of the system. The exact device on which Redis runs is not important, but the interconnection should be fast enough to allow uncompressed frames to be transmitted at the correct frame rate. Usually, Gigabit ethernet is fine, but Wifi won't be sufficient.

Hics is designed to control a pushbroom hyperspectral camera system, which generally consists of:

- Camera controls (integration time, frame rate)
- Framegrabber (captures each frame coming from the camera)
- Scanner (moves the camera, the sample, or a mirror in order to get a 2D image
- Focus (selection of the depth of sharpness)
- Shutter (controls whether the light reaches the sensor or not)

The exact way these components are connected to the hics system depend on the hardware. As an example, we present the different commands that should be run in order to use the setup used in chapter 4.

- `python3 -m hics.hardware.specimswir.camera` (on the server; camera control)
- `python3 -m hics.hardware.specimswir.scanner` (on the server; mirror scanner control)
- `python3 -m hics.hardware.specimswir.framegrabber` (on the server; frame grabber)
- `NiRawStreamer.exe img0 192.168.1.2 1234` (on the framegrabber; capture and send frames to the framegrabber. `img0` is the name of the NI interface)
- `python3 -m hics.hardware.ev3.ev3focus --redis redis://192.168.1.2` (on LEGO ev3 brick; focus control)
- `python3 -m hics.hardware.ev3.ev3rotater --redis redis://192.168.1.2` (on LEGO ev3 brick; rotating device)
- `python2 -m hics.hardware.webcam --redis redis://192.168.1.2` (on the laptop; webcam capture)
- `python3 -m hics.daemon.frameconverter` (on the server; basic correction for preview),
- `python3 -m hics.gui` (on the laptop; graphical user interface)
- `python3 -m hics.plugin.photogrammetry --redis redis://192.168.1.2` (on the laptop; capture data from webcam)
- `python3 -m hics.plugin.record --redis redis://192.168.1.2` (on the laptop; record hyperspectral frames to file)
- `python3 -m hics.plugin.autofocus --redis redis://192.168.1.2` (on any system; autofocus plugin)
- `python3 -m hics.plugin.labelprint --redis redis://192.168.1.2` (on any system; plugin to print labels)
- `python3 -m hics.plugin.autoscan --redis redis://192.168.1.2` (on any system; plugin to automate the plugins for library scanning)

As we can see, even though the command lines are quite simple (they mostly consist in indicating the redis server URL), there are numerous components. To allow simple testing, a hyperspectral camera simulator has been written. The simplest setup is to have a Redis server on the local computer, and run the following commands:

- `python3 -m hics.hardware.simulator --datafile datafile.scan` (simulate a hyperspectral camera using data from `datafile.scan`),
- `python3 -m hics.daemon.frameconverter` (basic correction for preview),
- `python3 -m hics.gui` (graphical user interface).

The simulator emulates the camera, the frame grabber, the scanner, the shutter and focus. This is in generally sufficient to test the graphical user interface and plugins.

A.2 Capturing data

Usually, the process for scanning a sample is the following:

1. Define the bounds for the scanner
2. Set the focus
3. Choose an integration time
4. Capture dark frames, and scan the sample

The specific setup is out of the scope of this user manual, but is instead discussed in chapter 2. We therefore consider that the hardware is functional (and all related components are running), or that the simulator is used. The only non-hardware component required is `hics.daemon.frameconverter` to do basic dark frame subtraction for the graphical user interface. It is generally run on the same computer as the Redis server.

First, launch the Hyperspectral Imaging Control System (HICS) graphical user interface (`python3 -m hics.gui`). If everything goes well, an empty window is shown. If the redis server is not reachable, the user will be asked to provide the Redis URL (it can also be changed using the `File / Settings` menu. Usually, one would like to have at least the camera settings, the scanner settings, the camera live output, and the waterfall plot displayed (as in fig. A.1). This can be done using the `Window` menu.

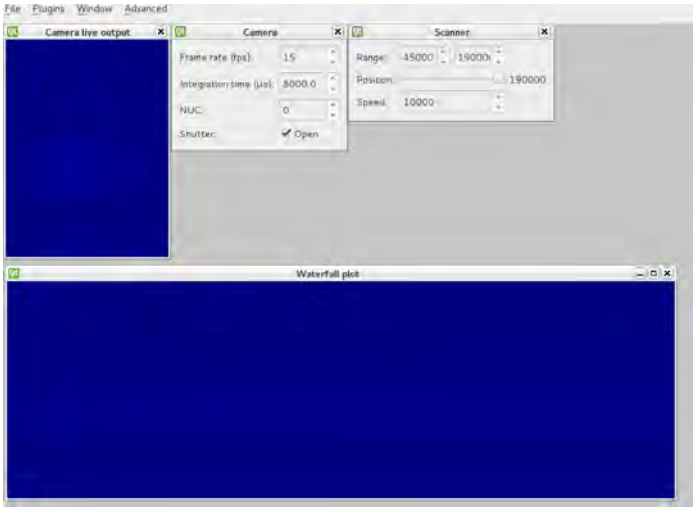


Figure A.1: HICS GUI, with the common windows displayed.

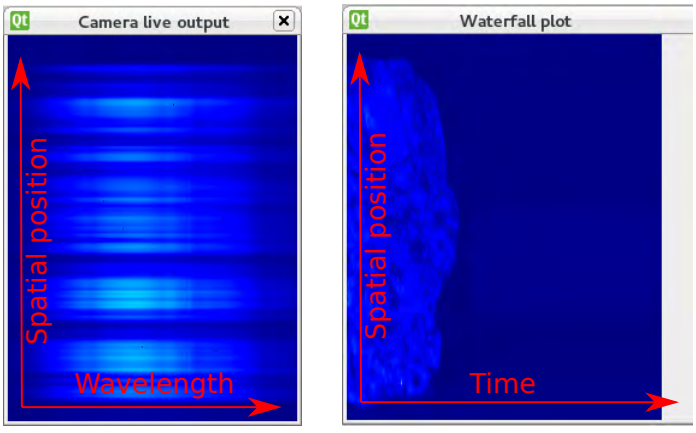


Figure A.2: On the left, the `Camera live output` window shows what is captured by the sensor. The horizontal dimension is spectral, the vertical dimension is spatial. Each captured frame is averaged spectrally, and added to the left of the `waterfall plot`, enabling the operator to see a crude 2D image.

There are a few additional functions to the interface. One can click on the camera live output to get a magnitude plot (right button) or a spectrum

plot (left button) for a given position. Also, the currently active plot window (or the camera live output) can be exported to Python script by using the `File / Export data` menu entry.

A.2.1 Estimating the scanner parameters

First, it is convenient to have the correct parameters for the scanner before working on any other settings, as it avoids getting "lost" is some unknown area.

To do so, the simplest method is to begin by settings convenient parameters:

- A human adapted frame rate (e.g. 25 fps)
- A large integration time (depends on the camera and the illumination, for example 10 000 µs)
- A fast scanning speed

Then, a large scanning range should be set. Using the position slider, the operator should try to find the beginning and the end of the region of interest, and adapt the bounds accordingly. Usually, the best approach is to follow an incremental process: first determine roughly the bounds, then lower the scanning speed, then adapt the bounds to increase their precision.

Usually, it is better to leave a few percent margin on both sides, to avoid accidental loss of the data on the sides, as the margin pixels can be masked easily in the data processing steps. On the other hand, it is not advised to have huge margins, since this will increase the size of the captured data.

A.2.2 Focus

Once the scanning bounds are set, it is required to focus the camera in order to get sharp pictures. If the setup supports motorized focus control, it is possible to control it using the window depicted in figure A.3, otherwise it has to be done manually by the operator by turning the focus rig.

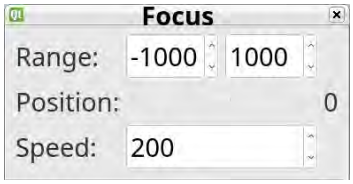


Figure A.3: Window to control the focus.

There are two plugins to help the operator to set the focus, both using contrast based methods. The first one is the manual focus plugin, which helps the operator to focus the image, which works both with manual and motorized focus. The second one is the automatic focus plugin. It requires a motorized focus. Their usage is otherwise similar.

The manual focus plugin can be launched using `python3 -m hics.plugin.focushelper`, and the automatic one using `python3 -m hics.plugin.focushelper`.

The first thing to do is to decide which region should be used for focus. If there is no specific region of interest, this should be the middle of the sample. First, move the position of the scanner to this position, using the position slider. Then, launch the `Manual focus` or the `Automatic focus` plugin using the `Plugins` menu. The automatic focus will ask for the percentage of the frame which should be used for contrast computation, while the manual one will use the full frame.

A new window will open, with a live plot of the contrast of the current image returned by the camera. For the automatic plugin, it is only for monitoring and everything will happen automatically. For the manual focus, the operator then turn the focus rig, in order to maximize the contrast shown on the plot. Usually the simplest approach is to turn the rig in one direction. If the contrast decreases, then the direction should be changed, otherwise just continue till a maximum is reached, and the values start decreasing again (at that time, the rig should be turned back a little). This is exactly the strategy that is implemented in the automatic plugin. Note that the window showing the contrast can be resized to improve its visibility.

Once the focusing is done, the window can be closed. If the image focus was changed a lot, it may be required to adapt the scanning area, as changing the focus may have a zooming effect.

A.2.3 Integration time

Finally, the last step is to define a good integration time (exposure). This is not strictly required as it is possible to capture multiple images, and then to merge them using HDR techniques, but it helps to have a starting point. The software is provided with the `Auto-exposure plugin`, and can be started using `python3 -m hics.plugin.autoexposure`, and then called using the `Plugins` menu.

It requires 3 parameters, the percentile (p), the lowest acceptable magnitude (M_{min}) and the highest acceptable magnitude (M_{max}). It will do multiple acquisition of the image within the scanning range, and will adapt the integration time, such that the p-th percentile of the captured image is between M_{min} and M_{max}. This percentile is used to avoid the few extreme

values that can arise (like a bad pixel). The default values ($p = 0.99$, $M_{min} = 0.7$, $M_{max} = 0.9$) are usually fine.

A.2.4 Other parameters

It is usually better to disable the camera non-uniformity correction by setting NUC to 0. To update the dark frame subtraction to the new parameters, one should toggle the shutter button in order to capture a few dark frames. The dark frame subtraction for the preview will be automatically adapted.

A.2.5 Scanning an image using the `record` plugin

There is a huge variety of type of scans that can be made (for example, HDR scans, time lapse, etc.). The most often used methods are implement in the `Record` plugin, which can be run using `python3 -m hics.plugin.recordscan`. Users requiring more peculiar acquisition types should implement them as plugins. This is not covered in this user manual.

A white reference should be put in the scanning range, and the scanner should be set to this position using the position slider. Then the `Record` plugin should be launched using the `Plugins` menu, and its window should appear (fig. A.4).

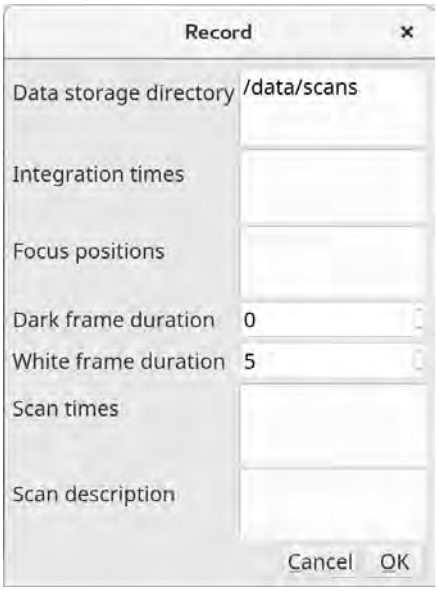


Figure A.4: The record plugin

The output directory can be specified using the `Data storage directory` field. A file will be created based on the current date and time. Depending on the other parameters specified, it can contain one or more scans. The parameters of the scans can be specified using the `Integration times`, `Focus positions` and `Scan times` fields. Each of these fields can contain either no value (will use the current value of the camera), or a comma separated list of ranges, using the following format:

- A single value (ex: `1234`)
- A range (`10:15` is equivalent to `10, 11, 12, 13, 14`)
- A range with step size (10:20:2 is equivalent to `10, 12, 14, 16, 18`)

`Scan times` can be used for a time lapse. For example specifying `Scan times = 0:3601:60` will make a capture each minute for one hour. Each of these captures will consist in possibly multiple scans for each combination of `Integration times` and `Focus positions`. Specifying multiple integration times can be useful if the sample has a large difference in luminosity between bright and dark areas. Multiple focus positions are only useful if the sample has a complex shape, in order to do focus stacking at a later stage.

`Dark frame duration` is the duration of the capture of dark frames before and after each scan, in order to estimate the average noise and variance of the dark current. To capture only a single frame, it is possible to set the value 0. `White frame duration` is the duration of the scan of the white frames. In addition to its usefulness to compute reflectance, white frames are used by calibration to estimate the noise pattern of the sensor and therefore to detect bad pixels.

Finally, `Scan description` is a field which can be freely used to write the description of the scan, this text is embedded in the scan file.

Once everything is set up, click the `OK` button, and the scan(s) will be automatically performed. The different windows can be used to monitor the status of the camera and the data captured.

A.3 Data processing

As many different data processing possibilities exist, we will show only a simple one, which should be generally sufficient. At every step, the file is a `mmappickle` [76] dictionary file. It can be loaded by the viewer, even if the processing is not finished. It is for example possible to display the contents of the file while the scans are being made. If `mmappickle` is not available, it can be loaded using `pickle.load`, at the expense of higher memory usage. At any steps, at least the following keys are available:

- `description`: contains a text description of the contents of the file. It is up to the user to provide a meaningful description.
- `wavelengths`: list or numpy array of the wavelengths corresponding to the bands
- `processing_steps`: list of the processing steps applied to the file. It is not present for files containing raw data only.

It is possible to list the contents of such files on the command line using the following command `python3 -m hics.datafile.ls`.

The file obtained when using the `Record` plugin contains the following keys (`#####` is the scan id, all hyperspectral data is in digital numbers):

- `scan-#####`: contains the data captured when doing the scan.
- `scan-#####-d0`, `scan-#####-d1`: dark frames captured before and after the scan
- `scan-#####-p`: properties of the scan (integration time, scanner positions, timestamps)
- `white-#####`: contains the white frame data.
- `white-#####-d0`, `white-#####-d1`: dark frames captured before and after the white frames
- `white-#####-p`: properties of the white frame scan (integration time, scanner positions, timestamps)

A.3.1 Removing bad pixels and computing reflectance

This step is done using the `hics.datafile.calibrate` tool. Its options can be seen using `python3 -m hics.datafile.calibrate --help`. For basic use, the default parameters are usually good enough, and it is sufficient to run `python3 -m hics.datafile.calibrate --input <filename.scan> --output <filename.calibrated>`.

The file obtained contains the following keys (`#####` is the scan id):

- `scan-#####`: contains the data captured when doing the scan, with dark frame removed and bad pixels masked. Units are DN μs^{-1}.
- `scan-#####-p`: properties of the scan (integration time, scanner positions, timestamps)
- `refl-#####`: contains the reflectance data.
- `refl-#####-var`: contains an estimate of the variance of the reflectance data.

- `refl-#####-w`: contains the white frame used to compute reflectance data.
- `refl-#####-p`: properties of the scan (integration time, scanner positions, timestamps)

A.3.2 Creating a HDR file

To merge the different scans, `hics.datafile.merge` can be used. The number of valid information required per pixel can be specified using the `--required_images` parameter. At least one image is required to get the value of the pixels, but if enough scans were made is it advised to increase this value in order to get a better quality HDR image. An example command line would be: `python3 -m hics.datafile.merge --input <filename.calibrated> --output <filename.hdr> --required_images 2 --clean`. The `--clean` parameter removes all intermediate data (which can be large) from the HDR file.

The file obtained contains the following keys:

- `hdr`: contains the HDR image.
- `hdr-var`: contains an estimate of the variance.

A.3.3 Creating and applying a mask

A mask can be created by any image manipulation software. It consists in a black and white image with same dimensions as the hyperspectral image, where white pixels correspond to pixels that should be kept.

The image can be exported as a grayscale `png` using: `python3 -m hics.datafile.maskdata --input <filename.hdr> --mask <filename.png>`.

Then, the image can be edited in order to create a mask such as the one shown in figure A.5. The mask can be applied using: `python3 -m hics.datafile.maskdata --input <filename.hdr> --mask <filename.png> --output <filename.mhdr>`. The output file contains the same keys as the HDR file.

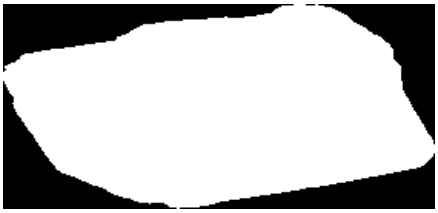


Figure A.5: An example mask for a hyperspectral image.

A.4 Data viewer

It can be useful to visualize a data file at any step, for reasons such as ensuring that the data quality is sufficient, or to do some quick analysis. For this purpose, a simple viewer has been developed. It can be launched using `python3 -m hics.viewer <filename>`, where `<filename>` is any hyperspectral data file produced. Figure A.6 shows an example view. The window is divided as follows:

- The list on the left shows the defined keys in the file.
- If the current key contains hyperspectral data, the top right area will contain 2D data, and the bottom right will contain spectra.
- If the current key contains other type of data, the right area shows directly the content of the key.

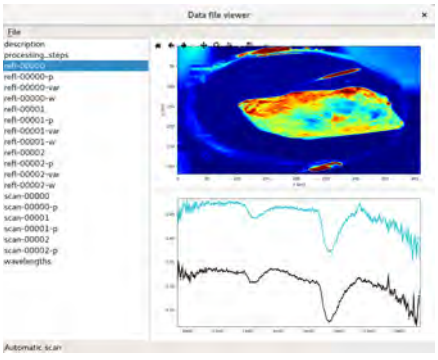


Figure A.6: The data viewer window, showing a calibrated file.

For hyperspectral data, the 2D data part will show the average intensity of each pixel. The spectra for each point in the hyperspectral data can be

seen by moving the mouse around in the 2D image. It is also possible to add points for which the spectra is always displayed (great for comparisons), by clicking with the right button in the 2D image, and selecting `Add point`. An example is shown in figure A.7.

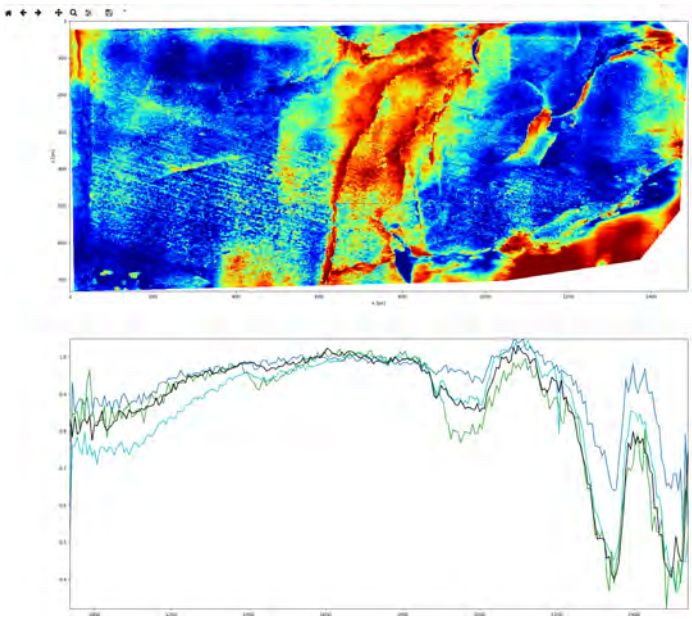


Figure A.7: The data viewer window, with three data points selected. The spectrum in black correspond to the mouse cursor.

It is possible to change the bands displayed by right clicking on the spectrum displayed, and mapping a specific band to a given color. This is very useful to compare the relative intensity between bands (fig. A.8.

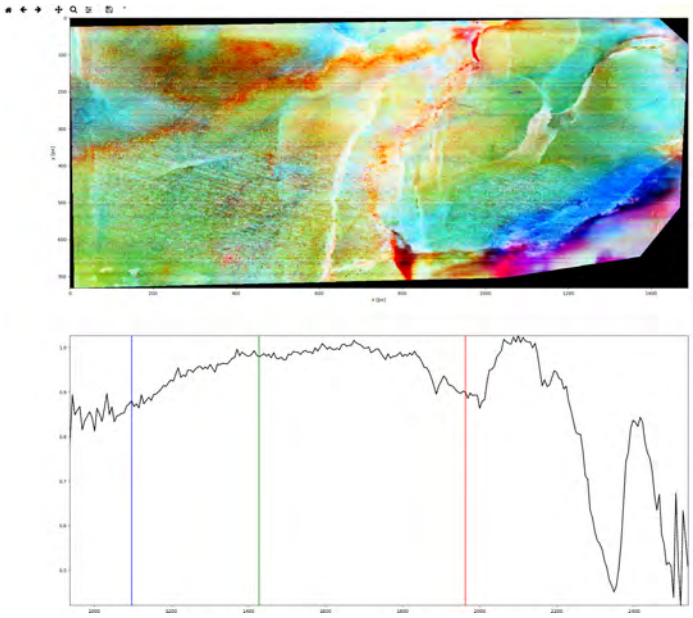


Figure A.8: The data viewer window, with 3 bands selected.

The colormap can be changed by using the ⚟ button. For each of the colorband, a color-keyed histogram of the band is shown, and the curve used to map the values to the color map. The curve can be edited, using the left mouse button to add or move a point, and the right one to remove one. The results can be seen directly on the other window (fig. A.9).

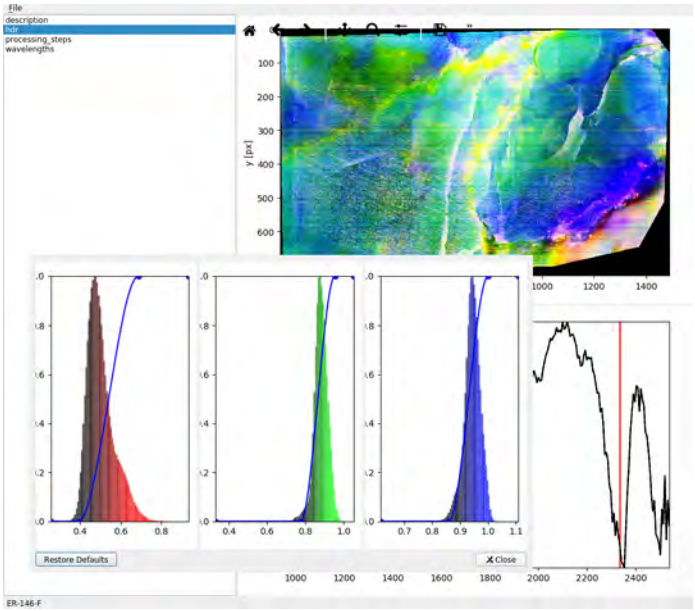


Figure A.9: The data viewer window, while the colormaps are being edited.

It is possible to save the current view state, using the `File -> Export` menu entry, in order to re-use it later. Indeed, it may require some time to get the "perfect" visualization settings, so it can be useful to store it for later use (for example when discussing results with other people). A saved view state can be reloaded using `python3 -m hics.viewer <data filename> <viewstate filename>`.

Appendix B

Bentonite surface water content determination using SWIR hyperspectral imaging

This chapter has been submitted as a technical report for NAGRA, under the title **AN 15-715 - FEBEX-DP Hyperspectral Scanning**, *on August 28, 2015.*

It is provided here, as the method was shown to be working.

Abstract

This report studies the feasibility of using a SWIR hyperspectral camera to measure the near-surface water content of bentonite.

We show that this technology is indeed suitable for this task and present a workflow to estimate the volumetric water content from a given spectrum of a pixel.

The workflow was then applied to hyperspectral scans of bentonite slice 22 of FEBEX-DP (fig B.1).

The report is structured in the following way:

- Part 1 gives a brief overview of the scanned data and quickly evaluates the quality of the calibration
- Part 2 describes a small laboratory measurement using a bentonite sample resulting in a method of identifying the water content of bentonite.

- Part 3 shows the results on the performed scans showed in part 1 using this method.
- Part 4 concludes.

Notions

Throughout this document, we will use the following notations:

- x, y position (in pixel)
- λ wavelength (in nanometers)
- t time (in second)
- $m(x, y, \lambda)$ is the calibrated intensity at position (x, y) at wavelength λ
- For any function, parameter is omitted when it doesn't vary (i.e. we do not write x when we work on a vertical line)
- If a parameter varies, and is omitted, we instead mean the value evaluated at all omitted parameters values. For example: $m(x, y)$ is the whole spectrum at the point (x, y) (which can be seen as a function $s(\lambda) = m(x, y, \lambda)$, with x,y fixed)

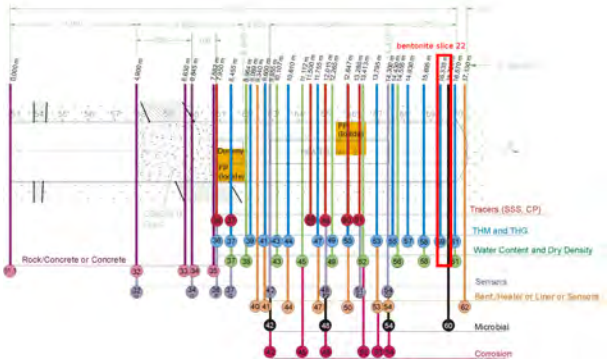


Figure B.1: Image of the FEBEX-DP dismantling section layout, indicating section 22 which was part of the dismantling sampling section 56.

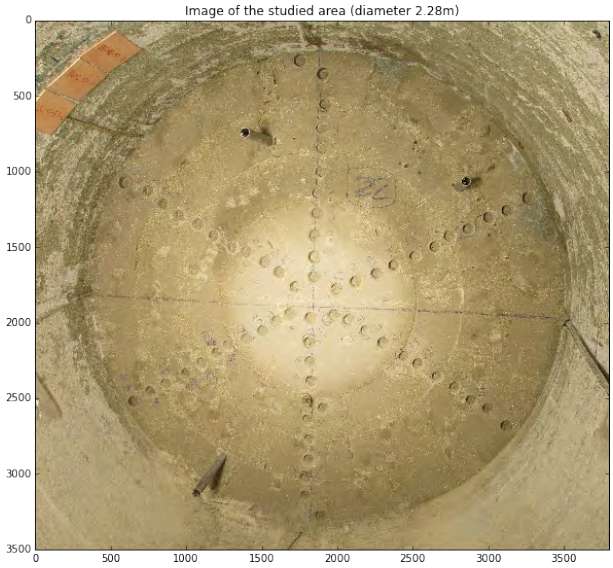


Figure B.2: RGB image of the investigated area (bentonite slice 22).

B.1 Scanned data and calibration

First, the results of the calibration process will be presented, then the scans, and finally a quick analysis will be made.

B.1.1 Experimental setup and calibration process

We use a SPECIM SWIR hyperspectral camera which is able to measure 256 wavelengths, from 937.06 to 2539.77nm, on a vertical line of 320 pixels. A mirror scanner, controlled by a stepper motor, is attached to the camera in order to be able to do 2D scans.

The scene/sample is illuminated using two halogen lamps that are connected through a Variac enabling us to change the light intensity.

As the non-uniformity correction provided by the manufacturer is not satisfactory (low quality, and unsuitable for long integration times), we applied a new calibration process developped in-house which is targeted at improving the smoothness of the output picture.

In addition, as the temperature was quite low in the underground lab, the aforementioned calibration step was not precise enough and we had to apply a second step basically consisting of adding an offset to each pixel in order to smooth the scanning results and mask invalid data.

B.1.2 Calibration results

To illustrate the calibration process, the fifth scan (fig. B.3) at 10ms integration time will be used[1]. The calibration of the camera was done on the 15th of August, very early in the morning, in order to have similar temperature conditions.

[1] the same steps were done on the other scans, but are omitted in this report

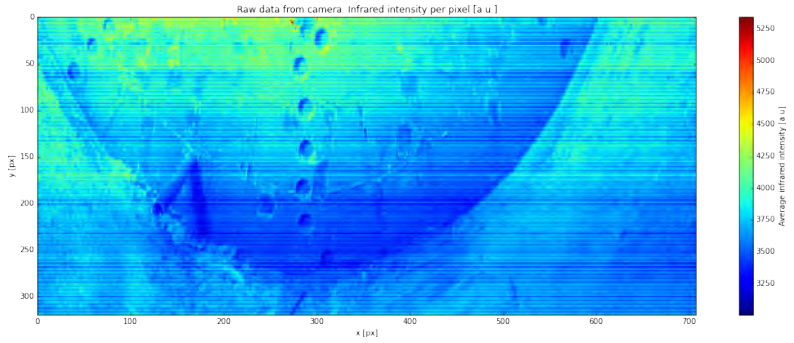


Figure B.3: Uncalibrated scan 5 (lower part of section 22). Axes are in pixels, the diameter of the tunnel is 2.28m.

As we can see, the results appear to be very noisy as there are a lot of pixels height horizontal stripes. Since the scanning direction is horizontal, this suggests that the pixels have different sensitivities creating these artifacts. Moreover, we can also see a difference in luminosity between the upper and lower half of the image which doesn't correspond to any observable difference in illumination.

After the aforementioned calibration, the results are significantly improved (fig. B.4):

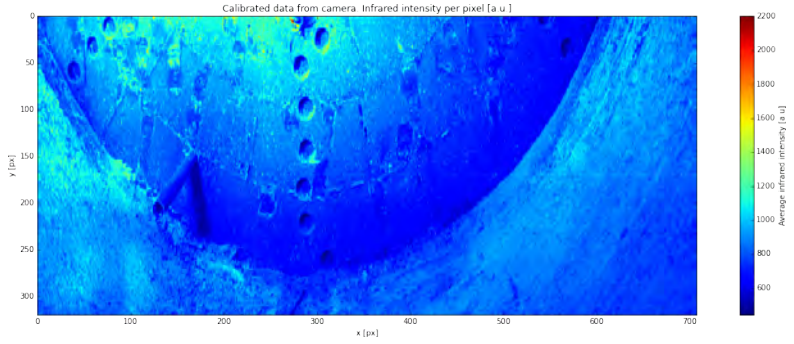


Figure B.4: Scan 5 after calibration step.

The results after calibration improved uniformity (much less horizontal stripes and there is no significant difference in intensity between the upper and lower part of the image). An analysis of the remaining horizontal stripes suggests that:

1. The temperature of the camera has a large impact on the sensor and should be taken into account.
2. There are "bad" pixels (which can be interpolated by using the pixels above and below)
3. There are issues with the calibration at some wavelengths:
    - at 962nm
    - between 1167nm and 1218nm
    - above 2410nm

A second step was applied, in order to fix these issues. As one can see in fig. B.6 this improves the quality of the results significantly. Note that there are no horizontal stripes left in the picture.

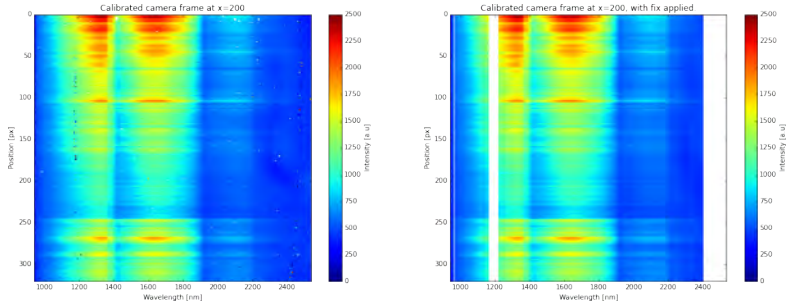


Figure B.5: Individual camera frame of scan 5 after calibration (on the left) and with fix applied and removal of bad data (on the right).

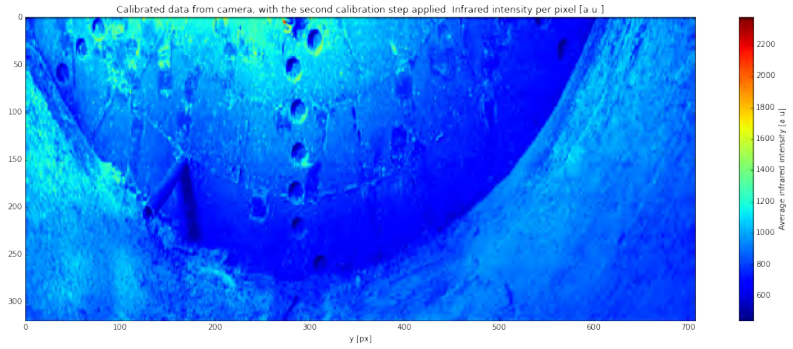


Figure B.6: Scan 5 after calibration and fix of bad data.

The fig. B.7 shows that the calibration reduced the variance for each wavelength. Reduction of variance is usually due to smoothing, in our case the calibration process removed the stripes and the bad pixels resulting in the smoothed data. This is confirmed by the fact that the large peaks of variances have been removed but the general shape was preserved (see fig. B.7).

The plot also suggests that the interesting spectral region are likely to be around the maximums at 1300nm and 1600nm.

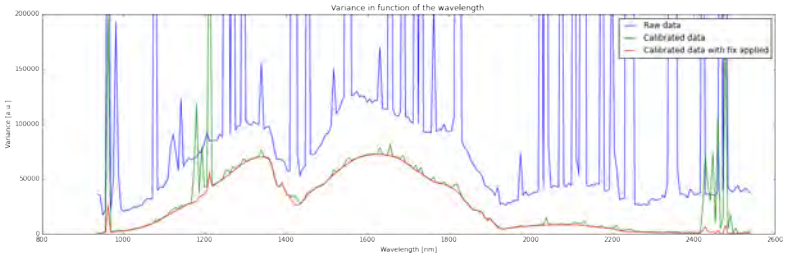


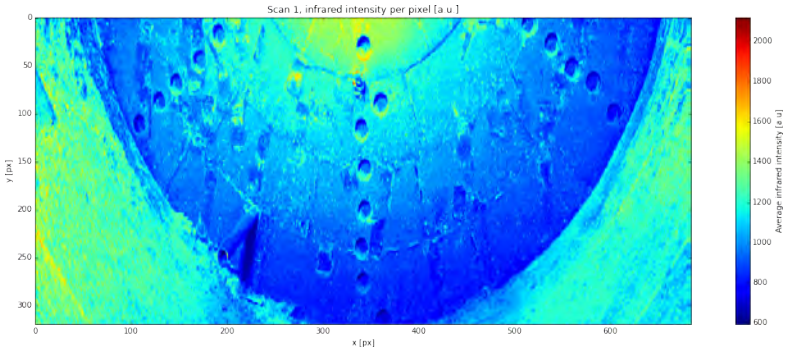
Figure B.7: Variance for each wavelength in scan 5.

B.1.3 Scans

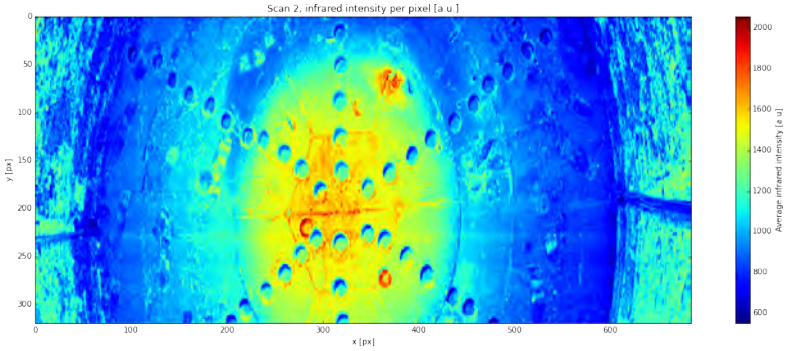
Five different scans were made:

- Scan 1 to 3 were made with low illumination.
- Scan 4 and 5 were made with higher illumination.

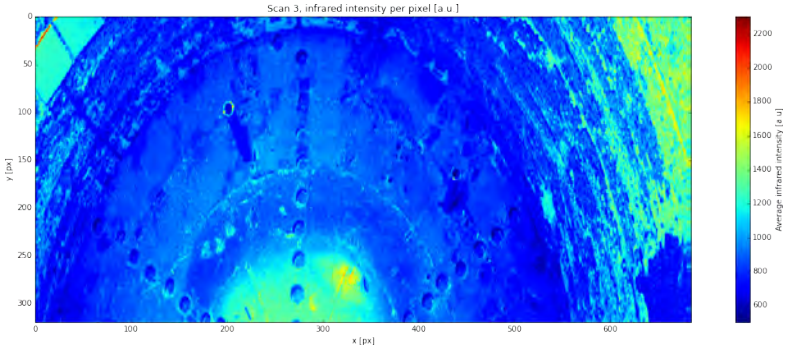
The average pixel intensity of the calibrated image of all these scan are shown in figure B.8.



(a) Scan 1

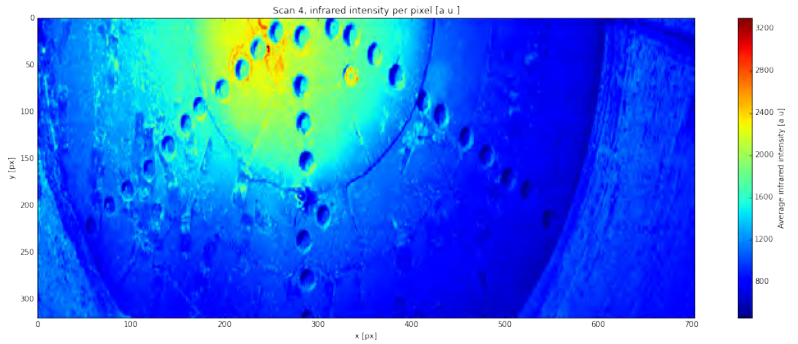


(b) Scan 2

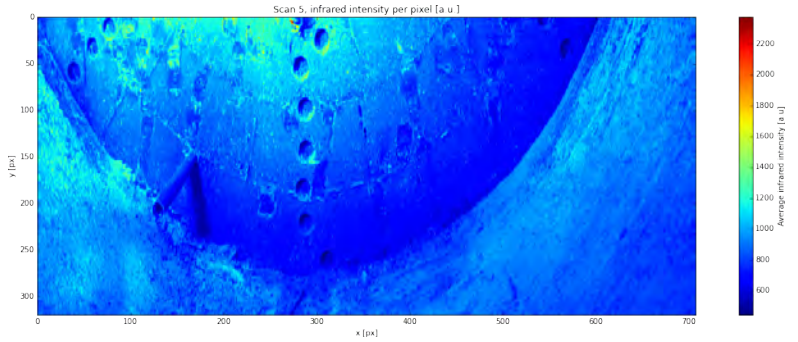


(c) Scan 3

Figure B.8: Average infrared pixel intensity



(a) Scan 4



(b) Scan 5

Figure 8 (cont.): Average infrared pixel intensity

B.1.4 Preliminary analysis

The standard minimum noise fraction transformation (MNF) didn't produce satisfactory results which is likely mainly due to the non-optimal calibration and the fact that the changes of near surface water content are smooth and continuous rather than abrupt. However, the variances shown in fig. B.7 suggest that modifications of the surface reflectance attributed to water content occur at wavelengths near 1300nm and 1600nm.

For instance, we can define arbitrary continuous relations like the following, shown in fig. 9:

$$p(x,y) = \frac{\int_{1573.2nm}^{1629.97nm} m(x,y,\lambda)d\lambda}{\int_{937.06nm}^{2539.77nm} m(x,y,\lambda)d\lambda}$$

where $m(x,y,\lambda)$ is the calibrated value. The visual structure (the same pattern as the one observed on site) strongly suggests that these wavelengths

are related to the water content. Therefore, an additional experiment was performed in the lab which will shortly be explained in part B.2.

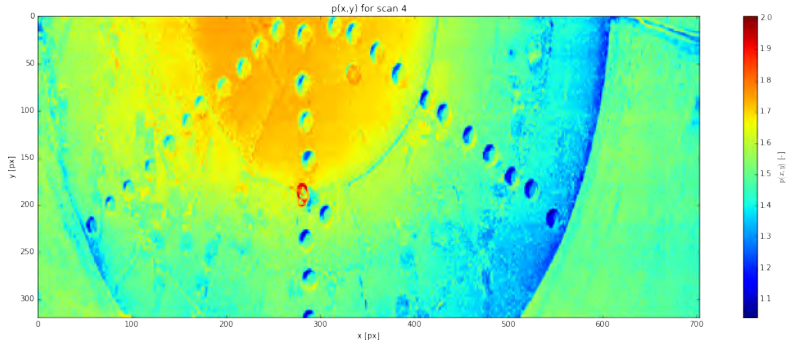


Figure 9: $p(x, y)$ for scan 4.

B.2 Lab experiment with bentonite

In order to gain a qualitative understanding of the behaviour of the spectrum of bentonite, part of a bentonite block was positioned in front of the hyperspectral camera in the lab, using halogen lamps both for lightning and heating. Then, at $t = 60s$, water was added. The idea was to have a relatively wet bentonite sample drying (out) quickly to be able to observe changes in the hyperspectral scans (disregarding possible other influences such as structural changes etc.).

Hyperspectral images were taken every 6s for 4 hours (the sample was drying during this time due to the heat from the halogen lamps). Figure 10 shows the setup and figure 11 the evolution of the spectrum during drying.

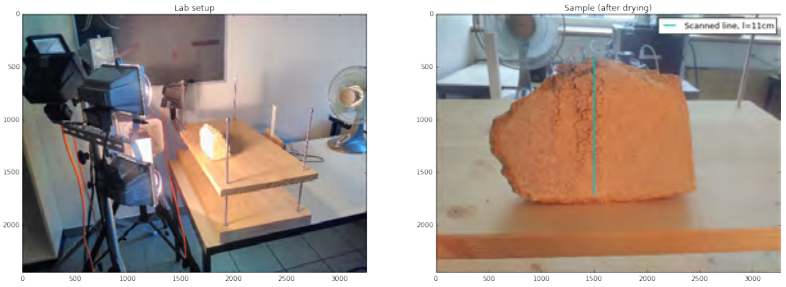


Figure 10: Setup of the experiment.

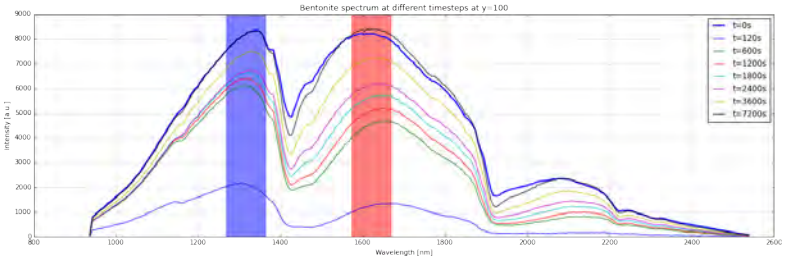


Figure 11: Evolution of the spectrum of a point during drying.

As we can see, the difference of the spectrum due to near-surface water content is prominent in the red (between 1573 and 1668nm) and in the blue (1269 to 1364nm) areas even though the exact water content were unknown in this simplifying lab test.

The main aim was to find a function f linking the measured spectrum $m(x, y)$ to the volumetric water content $v(x, y)$. To keep things simple, we will write f as a composition of two functions, $f = g \circ h$, where h is a function mapping the spectrum to a scalar, which is empirically defined, and g is fitted according to the measured water content of the cores from the FEBEX-DP experiment. Two different models were evaluated:

1. $h_1(x, y) = \frac{\int_{1563nm}^{1668nm} m(x,y,\lambda)d\lambda}{\int_{1269nm}^{1364nm} m(x,y,\lambda)d\lambda}$ (integral over the red area divided by the integral over the blue area)
2. $h_2(x, y) = \frac{\int_{1269nm}^{1364nm} m(x,y,\lambda)d\lambda}{\int_{937.06nm}^{2539.77nm} m(x,y,\lambda)d\lambda}$ (integral over the blue area divided by the integral over the whole spectrum)

Fig. 12 and 13 show these two functions for the lab experiment:

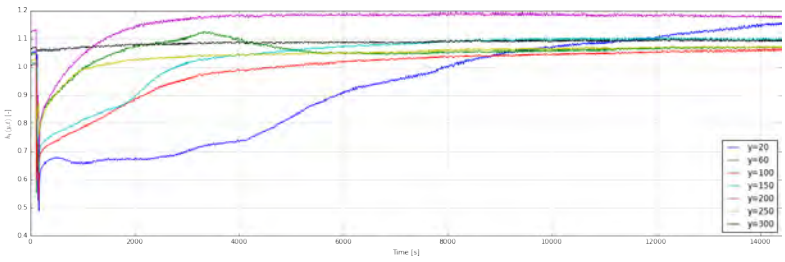


Figure 12: Evolution of h_1 at various positions.

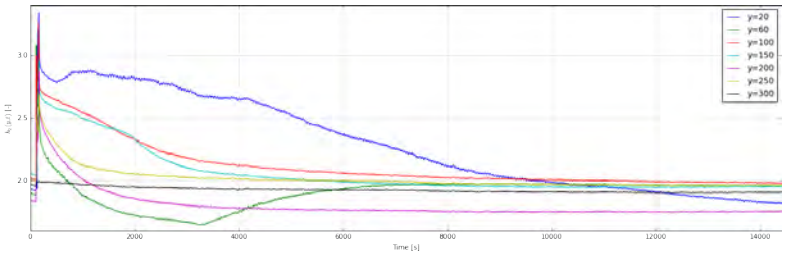


Figure 13: Evolution of h_2 at various positions.

As we can see, both functions capture the information about the drying bentonite in the lab experiment pretty well. Note however that h_2 is less sensitive to noise than h_1, since the divisor is taken over a wider spectral range.

In the next section, we will see how these functions correlate with the measured water content from the FEBEX-DP.

B.3 Correlation of spectral measurements and water content

The measured volumetric water content of cores taken from the FEBEX-DP slice 22 (compare fig B.1) is given in table B.1.

Table B.1: Volumetric water content measured for selected cores of slice 22 in the FEBEX-DP (original data from Villar, Maria & Iglesias, Rubén Javier (2016): FEBEX-DP Bentonite Onsite Analysis Report. Nagra Arbeitsbericht NAB 16-012.)

Bentonite core number	Volumetric water content [%]
12	28.7
13	27.3
14	26.1
15	23.2
16	24.7
23	28.6
24	26.2
25	25.8
26	24.9
27	25.8
28	24.2
29	23.5
30	21.7
31	20.5
32	19.1
45	28.1
46	26.0
47	25.2
48	23.5
49	23.1
50	22.8

Since the water content of the bentonite cores are known, it can be used to find g, and subsequently f.

Rectangular areas (patches) approximately in the center of the boreholes (see fig. 14) were defined in the images where water content is known from precise measurements.

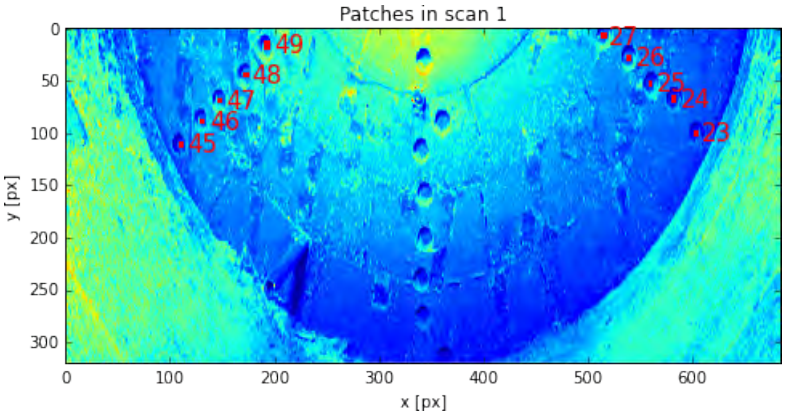


Figure 14: Patches in scan 1 of the investigated FEBEX-DP bentonite slice 22 (dismantling sampling section 56).

Therefore, we can compare the results of $t_1(m(x,y))$ and $t_2(m(x,y))$ for each patch with the water content of the corresponding core (fig. 15).

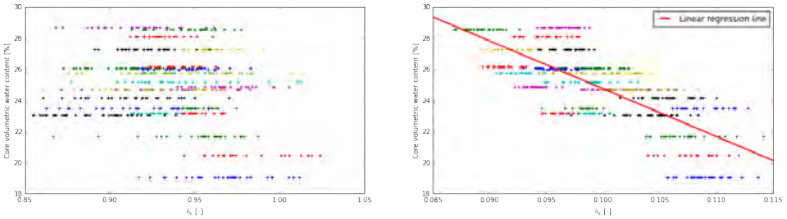


Figure 15: Water content in function of h_1 (left) resp. h_2 (right).

As we can see, h_1 does not give an satisfactory correlation (which is probably due to its high sensitivity to noise, as previously stated) whereas h_2 seems clearly related to the volumetric water content.

Since we do have only limited data points, only a linear fit was made:

$$v(x,y) = -307.14 \cdot \frac{\int_{1269nm}^{1364nm} m(x,y,\lambda)d\lambda}{\int_{937.06nm}^{2539.77nm} m(x,y,\lambda)d\lambda} + 55.48$$

Without further, more detailed lab tests and given the data used for the calibration and the experiment setup (e.g. the lighting conditions), the estimate is likely to be valid only for volumetric water contents between 21 and 27% and is not very precise (prediction seems to be ±2%). Figures 16

to 20 show the previous relation applied to the scans. Note that the value makes no sense for non-bentonite pixels, and for values out of the validity range.

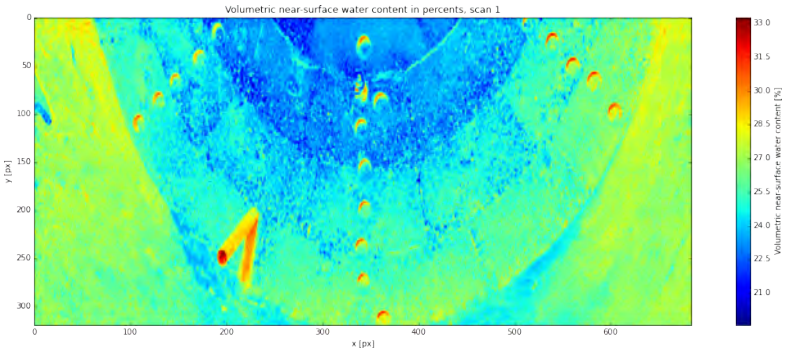


Figure 16: Volumetric near-surface water content in percent, scan 1

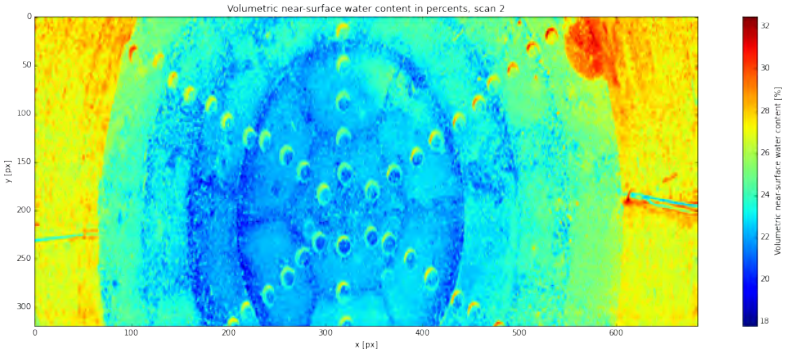


Figure 17: Volumetric near-surface water content in percent, scan 2

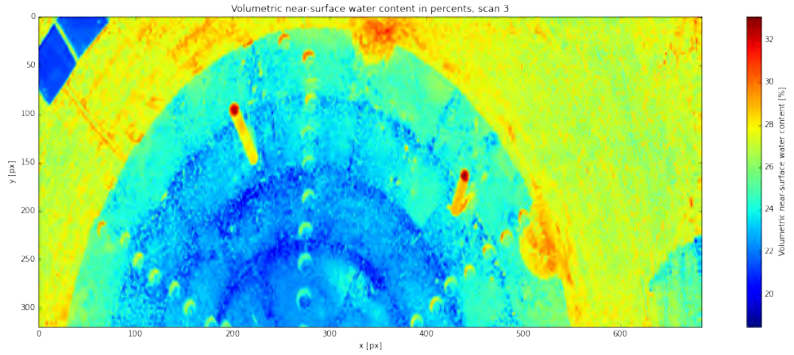


Figure 18: Volumetric near-surface water content in percent, scan 3

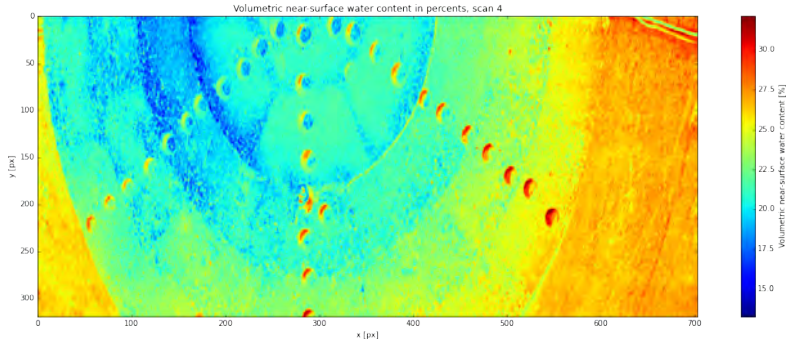


Figure 19: Volumetric near-surface water content in percent, scan 4

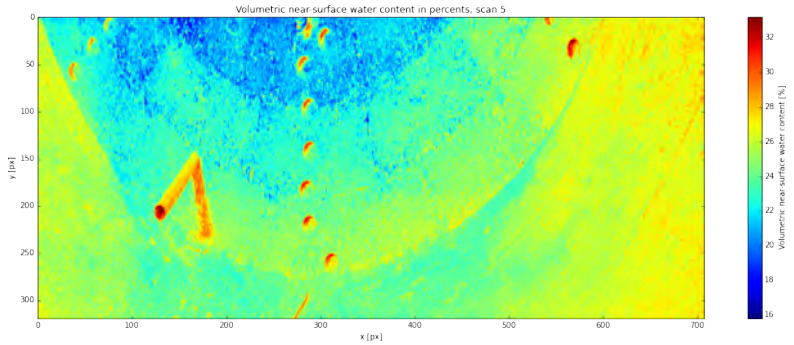


Figure 20: Volumetric near-surface water content in percent, scan 5

B.4 Conclusion

B.4.1 Achieved work

This report shows that hyperspectral imaging is suitable to determine near-surface water content of bentonite.

Five hyperspectral scans of slice 22 were made and, despite the calibration issues due to temperature, high quality data was obtained. An estimation of the relation between the spectrum and the water content could be established:

$$v(x,y) = -307.14 \cdot \frac{\int_{1269nm}^{1364nm} m(x,y,\lambda)d\lambda}{\int_{937.06nm}^{2539.77nm} m(x,y,\lambda)d\lambda} + 55.48$$

B.4.2 Further work

The relationship between the bentonite water content and its spectrum should be studied in more detail. In particular, the formula should be extended in order to take into account the spectrum of the light source, or equivalently to work using the reflectance of the sample instead of using the spectrum of the light emitted back. Also, samples with known water content should be scanned in optimal conditions in order to have less measurement noise than in the underground lab.

Moreover, this would allow:

- an increase of the validity range of the formula
- a higher order fit
- a better precision in the values, since the formula would be calibrated on low noise data.

On the engineering side, the influence of the temperature on the measurement should be better characterized to avoid the extra post-calibration step which is not as rigorous as the calibration process.

Bibliography

[1] Austin Richards. *Alien Vision: Exploring the Electromagnetic Spectrum with Imaging Technology, Second Edition*, volume 9. SPIE press Bellingham and Washington, DC, 2011.

[2] *Sources and Characteristics of Remote Sensing Image Data*, pages 1–25. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-29711-6. doi: 10.1007/3-540-29711-1_1. URL `https://doi.org/10.1007/3-540-29711-1_1`.

[3] J_W Rouse Jr, RH Haas, JA Schell, and DW Deering. Monitoring vegetation systems in the Great Plains with ERTS. 1974.

[4] Susan K. Jenson and Frederick A. Waltz. Principal components analysis and canonical analysis in remote sensing. 1:337–348, 01 1979.

[5] Andrew A Green, Mark Berman, Paul Switzer, and Maurice D Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on geoscience and remote sensing*, 26(1):65–74, 1988.

[6] Ashbindu Singh and Andrew Harrison. Standardized principal components. *International Journal of Remote Sensing*, 6(6):883–896, 1985.

[7] Christopher Gordon. A generalization of the maximum noise fraction transform. *arXiv preprint physics/0009019*, 2000.

[8] Laurens Van Der Maaten, Eric Postma, and Jaap Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10:66–71, 2009.

[9] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[10] F.A. Kruse, A.B. Lefkoff, J.W. Boardman, K.B. Heidebrecht, A.T. Shapiro, P.J. Barloon, and A.F.H. Goetz. The spectral image processing system (SIPS)—interactive visualization and analysis of imaging

spectrometer data. *Remote Sensing of Environment*, 44(2):145 – 163, 1993. ISSN 0034-4257. doi: http://dx.doi.org/10.1016/0034-4257(93)90013-N. URL http://www.sciencedirect.com/science/article/pii/003442579390013N.

[11] Bing Zhang, Shanshan Li, Changshan Wu, Lianru Gao, Wenjuan Zhang, and Man Peng. A neighbourhood-constrained k-means approach to classify very high spatial resolution hyperspectral imagery. *Remote sensing letters*, 4(2):161–170, 2013.

[12] R. Heylen, M. Parente, and P. Gader. A review of nonlinear hyperspectral unmixing methods. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 7(6):1844–1868, June 2014. ISSN 1939-1404. doi: 10.1109/JSTARS.2014.2320576.

[13] Nirmal Keshava and John F Mustard. Spectral unmixing. *IEEE signal processing magazine*, 19(1):44–57, 2002.

[14] Robert S. Rand, Ronald G. Resmini, and David W. Allen. Characterizing intimate mixtures of materials in hyperspectral imagery with albedo-based and kernel-based approaches, 2015. URL http://dx.doi.org/10.1117/12.2190067.

[15] Bruce Hapke. *Theory of Reflectance and Emittance Spectroscopy*. Cambridge University Press, 2 edition, 2012. doi: 10.1017/CBO9781139025683.

[16] Y. Shkuratov, V. Kaydash, V. Korokhin, Y. Velikodsky, D. Petrov, E. Zubko, D. Stankevich, and G. Videen. A critical assessment of the hapke photometric model. *Journal of Quantitative Spectroscopy and Radiative Transfer*, 113(18):2431 – 2456, 2012. ISSN 0022-4073. doi: https://doi.org/10.1016/j.jqsrt.2012.04.010. URL http://www.sciencedirect.com/science/article/pii/S0022407312001926. Electromagnetic and Light Scattering by nonspherical particles XIII.

[17] Y. Yuan, Y. Feng, and X. Lu. Projection-based nmf for hyperspectral unmixing. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(6):2632–2643, June 2015. ISSN 1939-1404. doi: 10.1109/JSTARS.2015.2427656.

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436, 2015.

[19] I.A Basheer and M Hajmeer. Artificial neural networks: fundamentals, computing, design, and application. *Journal of Microbiological Methods*, 43(1):3 – 31, 2000. ISSN 0167-7012. doi: https://doi.org/10.

1016/S0167-7012(00)00201-3. URL http://www.sciencedirect.com/science/article/pii/S0167701200002013. Neural Computting in Micrbiology.

[20] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6): 386, 1958.

[21] Olivier Temam. The rebirth of neural networks. In *International Symposium on Computer Architecture*, 2010.

[22] Nicholas C Thomas. The early history of spectroscopy. *Journal of chemical education*, 68(8):631, 1991.

[23] Gregg Vane, Alexander FH Goetz, and John B Wellman. Airborne imaging spectrometer: A new tool for remote sensing. *IEEE Transactions on Geoscience and Remote Sensing*, (6):546–549, 1984.

[24] K.Ya. Kondratyev, O.B. Vassilyev, A.A. Grigoryev, and G.A. Ivanian. An analysis of the Earth's Resources Satellite (ERTS-1) data. *Remote Sensing of Environment*, 2:273 – 283, 1971–1973. ISSN 0034-4257. doi: http://doi.org/10.1016/0034-4257(71)90100-3. URL http://www.sciencedirect.com/science/article/pii/0034425771901003.

[25] Alexander F.H. Goetz, Gregg Vane, Jerry E. Solomon, and Barrett N. Rock. Imaging Spectrometry for Earth Remote Sensing. *Science*, 228 (4704):1147–1153, 1985. ISSN 0036-8075. doi: 10.1126/science.228.4704.1147. URL http://science.sciencemag.org/content/228/4704/1147.

[26] John H Gruninger, Anthony J Ratkowski, and Michael L Hoke. The sequential maximum angle convex cone (SMACC) endmember model. In *Defense and Security*, pages 1–14. International Society for Optics and Photonics, 2004.

[27] Farzeen Chaudhry, Chao-Cheng Wu, Weimin Liu, Chein-I Chang, and Antonio Plaza. Pixel purity index-based algorithms for endmember extraction from hyperspectral imagery. *Recent advances in hyperspectral signal and image processing*, 37(2):29–62, 2006.

[28] Piercesare Secchi, Simone Vantini, and Valeria Vitelli. Bagging voronoi classifiers for clustering spatial functional data. *International Journal of Applied Earth Observation and Geoinformation*, 22:53–64, 2013.

[29] Fred A Kruse. Mapping surface mineralogy using imaging spectrometry. *Geomorphology*, 137(1):41–56, 2012.

[30] Tobias H Kurz, Julie Dewit, Simon J Buckley, John B Thurmond, David W Hunt, and Rudy Swennen. Hyperspectral image analysis of different carbonate lithologies (limestone, karst and hydrothermal dolomites): the Pozalagua Quarry case study (Cantabria, North-west Spain). *Sedimentology*, 59(2):623–645, 2012.

[31] Da-Wen Sun, editor. *Hyperspectral Imaging for Food Quality Analysis and Control*. Academic Press, 2010. URL `http://www.sciencedirect.com/science/book/9780123747532`.

[32] Roberto Moscetti, Wouter Saeys, Janos C. Keresztes, Mohammad Goodarzi, Massimo Cecchini, Monarca Danilo, and Riccardo Massantini. Hazelnut Quality Sorting Using High Dynamic Range Short-Wave Infrared Hyperspectral Imaging. *Food and Bioprocess Technology*, 8 (7):1593–1604, 2015. ISSN 1935-5149. doi: 10.1007/s11947-015-1503-2. URL `http://dx.doi.org/10.1007/s11947-015-1503-2`.

[33] Guolan Lu and Baowei Fei. Medical hyperspectral imaging: a review. *Journal of Biomedical Optics*, 19(1):010901, 2014. doi: 10.1117/1.JBO.19.1.010901. URL `http://dx.doi.org/10.1117/1.JBO.19.1.010901`.

[34] Paul Geladi, Jim Burger, and Torbjörn Lestander. Hyperspectral imaging: calibration problems and solutions. *Chemometrics and Intelligent Laboratory Systems*, 72(2):209 – 217, 2004. ISSN 0169-7439. doi: https://doi.org/10.1016/j.chemolab.2004.01.023. URL `http://www.sciencedirect.com/science/article/pii/S0169743904000371`. Advances in Chromatography and Electrophoresis - Conferentia Chemometrica 2003, Budapest.

[35] Wojciech Matusik, Hanspeter Pfister, Matthew Brand, and Leonard McMillan. Efficient isotropic brdf measurement. 2003.

[36] MA Mohd Shahrimie, Puneet Mishra, Stien Mertens, Stijn Dhondt, Nathalie Wuyts, and Paul Scheunders. Modeling effects of illumination and plant geometry on leaf reflectance spectra in close-range hyperspectral imaging. In *Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), 2016 8th Workshop on*, pages 1–4. IEEE, 2016.

[37] Michael Goesele, Wolfgang Heidrich, and Hans-Peter Seidel. Entropy-based dark frame subtraction. In *PICS*, pages 293–298, 2001.

[38] Ralf Widenhorn, Morley M. Blouke, Alexander Weber, Armin Rest, and Erik Bodegom. Temperature dependence of dark current in a ccd, 2002. URL `http://dx.doi.org/10.1117/12.463446`.

[39] Ralf Widenhorn, Armin Rest, Morley M. Blouke, Richard L. Berry, and Erik Bodegom. Computation of dark frames in digital imagers, 2007. URL https://doi.org/10.1117/12.714784.

[40] Ralf Widenhorn, Ines Hartwig, Justin C Dunlap, and Erik Bodegom. Measurements of dark current in a ccd imager during light exposures, 2008. URL https://doi.org/10.1117/12.769082.

[41] Joseph Jablonski, Christopher Durell, Terrence Slonecker, Kwok Wong, Blair Simon, Andrew Eichelberger, and Jacob Osterberg. Best practices in passive remote sensing VNIR hyperspectral system hardware calibrations , 2016. URL http://dx.doi.org/10.1117/12.2224022.

[42] C.S. Bell. Contrast-based autofocus mechanism, December 8 1992. URL https://www.google.ch/patents/US5170202. US Patent 5,170,202.

[43] W. Lang. Method of and device for the automatic focusing of microscopes, March 20 1973. URL http://www.google.ch/patents/US3721759. US Patent 3,721,759.

[44] Frans C. A. Groen, Ian T. Young, and Guido Ligthart. A comparison of different focus functions for use in autofocus algorithms. *Cytometry*, 6(2):81–91, 1985. ISSN 1097-0320. doi: 10.1002/cyto.990060202. URL http://dx.doi.org/10.1002/cyto.990060202.

[45] Loren Shih. Autofocus survey: a comparison of algorithms, 2007. URL http://dx.doi.org/10.1117/12.705386.

[46] S. Mann and R.W. Picard. Being 'undigital' with digital cameras: Extending dyanmic range by combining differently exposed pictures. Technical Report 323, M.I.T. Media Lab Perceptual Computing Section, Boston, Massachusetts, 1994.

[47] P. E. Anuta. Spatial registration of multispectral and multitemporal digital imagery using fast fourier transform techniques. *IEEE Transactions on Geoscience Electronics*, 8(4):353–368, Oct 1970. ISSN 0018-9413. doi: 10.1109/TGE.1970.271435.

[48] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision*, pages 2564–2571, Nov 2011. doi: 10.1109/ICCV.2011.6126544.

[49] Frédéric Ratle, Gustavo Camps-Valls, and Jason Weston. Semisupervised neural networks for efficient hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 48(5):2271–2282, 2010.

[50] Konstantinos Makantasis, Konstantinos Karantzalos, Anastasios Doulamis, and Nikolaos Doulamis. Deep supervised learning for hyperspectral data classification through convolutional neural networks. In *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*, pages 4959–4962. IEEE, 2015.

[51] Wei Hu, Yangyu Huang, Li Wei, Fan Zhang, and Hengchao Li. Deep convolutional neural networks for hyperspectral image classification. *Journal of Sensors*, 2015, 2015.

[52] Heming Liang and Qi Li. Hyperspectral imagery classification using sparse representations of convolutional neural network features. *Remote Sensing*, 8(2):99, 2016.

[53] Gamal ElMasry, Ning Wang, and Clément Vigneault. Detecting chilling injury in red delicious apple using hyperspectral imaging and neural networks. *Postharvest biology and technology*, 52(1):1–8, 2009.

[54] Royston Goodacre, Rebecca Burton, Naheed Kaderbhai, Andrew M Woodward, Douglas B Kell, Paul J Rooney, et al. Rapid identification of urinary tract infection bacteria using hyperspectral whole-organism fingerprinting and artificial neural networks. *Microbiology*, 144(5): 1157–1170, 1998.

[55] Qiu-Xiang Yi, Jing-Feng Huang, Fu-Min Wang, Xiu-Zhen Wang, and Zhan-Yu Liu. Monitoring rice nitrogen status using hyperspectral reflectance and artificial neural network. *Environmental science & technology*, 41(19):6770–6775, 2007.

[56] Antonio Plaza, Jon Atli Benediktsson, Joseph W Boardman, Jason Brazile, Lorenzo Bruzzone, Gustavo Camps-Valls, Jocelyn Chanussot, Mathieu Fauvel, Paolo Gamba, Anthony Gualtieri, et al. Recent advances in techniques for hyperspectral image processing. *Remote sensing of environment*, 113:S110–S122, 2009.

[57] Volker Tresp, Subutai Ahmad, and Ralph Neuneier. Training neural networks with deficient data. In *Advances in neural information processing systems*, pages 128–135, 1994.

[58] Stéphane Dray and Julie Josse. Principal component analysis with missing values: a comparative survey of methods. *Plant Ecology*, 216 (5):657–667, 2015.

[59] Kiri Wagstaff. Clustering with missing values: No imputation required. In David Banks, Frederick R. McMorris, Phipps Arabie, and Wolfgang Gaul, editors, *Classification, Clustering, and Data Mining Applications*, pages 649–658, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-642-17103-1.

[60] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[61] Radomir S Stanković and Bogdan J Falkowski. The haar wavelet transform: its status and achievements. *Computers & Electrical Engineering*, 29(1):25–44, 2003.

[62] Piotr Porwik and Agnieszka Lisowska. The haar-wavelet transform in digital image processing: its status and achievements. *Machine graphics and vision*, 13(1/2):79–98, 2004.

[63] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[64] Freek D Van der Meer, Harald MA Van der Werff, Frank JA van Ruitenbeek, Chris A Hecker, Wim H Bakker, Marleen F Noomen, Mark van der Meijde, E John M Carranza, J Boudewijn de Smeth, and Tsehaie Woldai. Multi-and hyperspectral geologic remote sensing: A review. *International Journal of Applied Earth Observation and Geoinformation*, 14(1):112–128, 2012.

[65] Raymond F Kokaly, Roger N Clark, Gregg A Swayze, K Eric Livo, Todd M Hoefen, Neil C Pearson, Richard A Wise, William M Benzel, Heather A Lowers, Rhonda L Driscoll, et al. USGS spectral library version 7. Technical report, US Geological Survey, 2017.

[66] C Klein and CS Hurlbut Jr. Manual of Mineralogy (after JD Dana), revised 21st edn, 1999.

[67] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[68] openMVS. URL `https://github.com/cdcseacave/openMVS`.

[69] Laurent Fasnacht. Hyperspectral imaging system controller. `https://github.com/UniNE-CHYN/hics`, 2018.

[70] MCode Programming and Software Reference. URL `https://motion.schneider-electric.com/downloads/manuals/MCode.pdf`.

[71] Xclib; frame grabber programming library. URL `http://www.epixinc.com/products/xclib.htm`.

[72] Photonfocus user manual - register based ascii protocol. URL `http://www.photonfocus.com/fileadmin/web/manuals/MAN050_e_V1_2_ASCII_PROTOCOL.pdf`.

[73] ev3dev. URL `https://www.ev3dev.org/`.

[74] Laurent Fasnacht. Thorpy: Python library implementing thorlabs apt communication protocol, 2018. URL `https://github.com/UniNE-CHYN/thorpy`.

[75] Redis. URL `https://redis.io/`.

[76] Laurent Fasnacht. mmappickle: Python 3 module to store memory-mapped numpy array in pickle format. *Journal of Open Source Software*, 3(26):651:1–651:2, June 2018. ISSN 2475-9066. doi: https://doi.org/10.21105/joss.00651. URL `http://joss.theoj.org/papers/10.21105/joss.00651`.