Master's thesis

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

Mørenskog, Erik Rundhovde

Real-time hyperspectral sorting for industrial applications

A methodology for synchronising hyperspectral object classification with arbitrary sorting machines.

Master's thesis in ELSYS

Supervisor: Hernandez, Julio, Muri, Harald Ian, Hjelme, Dag Roar

July 2020

NTNU
Norwegian University of Science and Technology

Mørenskog, Erik Rundhovde

Real-time hyperspectral sorting for industrial applications

A methodology for synchronising hyperspectral object classification with arbitrary sorting machines.

Master's thesis in ELSYS
Supervisor: Hernandez, Julio, Muri, Harald Ian, Hjelme, Dag Roar
July 2020

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

NTNU
Norwegian University of
Science and Technology

Abstract

Hyperspectral imaging has proven to be a useful tool for remote sensing in lab, field, airborne and industry applications. The technique provides high spectral resolution in the visible and infrared spectrum, which is used for analysis and classification of substances. This thesis focuses on the industry application of sorting based on classification with hyperspectral imaging. Sorting out unwanted objects or defects can significantly increase efficiency in production lines.

The thesis explores how to make a classification module with a hyperspectral camera as the sensor and an external oscillator for synchronisation to third party sorting machines. The camera used is a Short Wave Infrared (SWIR) hyperspectral push-broom camera with spectral range 930-2500nm from HySpex. The performance and stability of the camera is verified through a test report review. The oscillator is a Frequency Divider from NEOLund, and the stability of the Frequency Divider is verified with an oscilloscope. The classification module is made to be integrable with any sorting system, for example, conveyor belts with pneumatic valves or stages with robotic arms. In this thesis, the classification module is presented in three parts: image acquisition and modelling, implementation of a general classification module, and testing with a specific integration.

As for all push-broom cameras, the samples are moved relative to the camera and scanned line by line during image acquisition. In this thesis, six samples are scanned: three pieces of wood and three pieces of rock. The samples are chosen to be easy to classify for the human eye so that the results can be verified manually. Two workflows are created to separate the samples from the background and classify them. Workflow 1 uses all of the 288 wavelengths the camera can supply, while workflow 2 limits the number of wavelengths to 56 with windowing. Windowing increases the maximum frequency of the camera and reduces the processing time of the workflow. The windowing is done based on information gathered when creating workflow 1. The separation of samples from the background is successful for workflow 1, and the classification model is accurate. The separation model in workflow 2 is noisier than in workflow 1, but still successful. The classification model is stronger than in workflow 1.

The proposed general classification module is coordinated based on collecting a set number of lines into easy to process frames. This collection allows the samples to be sorted based on a first-come, first sorted basis. The Breeze Runtime Software from Prediktera is used for real-time separation and classification of the samples. A program, named SynchSoftware, was developed as part of the thesis. It has a GUI that gives the user control over Breeze Runtime and the Frequency Divider. SynchSoftware also receives classified samples from Breeze Runtime, waits until all samples below a maximum size arrive and collects them into frames.

A test of a specific integration was performed with an FPGA to act as the sorting machine. The Frequency Divider controls the synchronisation through sending trigger pulses to the camera and event trigger pulses with 1/128 of the frequency to the FPGA. The FPGA then creates timestamps based on the event trigger pulses and sends them to SynchSoftware. SynchSoftware coordinates each timestamp to a frame and sends them as byte-string messages over TCP/IP to the FPGA. The synchronisation of the system relies on the stability of the Frequency Divider. The trigger signal was found to have a deviation of less than one-thousandth of its period. SynchSoftware was found to be limited to a camera frequency of below 125Hz. That limitation is due to SynchSoftware using much of its time on logging. The developed classification module consisting of the camera, Breeze Runtime, the Frequency Divider and SynchSoftware, can be integrated with any third party sorting machine.

Sammendrag

Hyperspektral avbildning har vist seg å være et nyttig verktøy for fjernmåling i laboratorie-, felt-, luftbårne og industrielle applikasjoner. Teknikken gir høy spektral oppløsning i det synlige og infrarøde spekteret, som kan brukes til analyse og klassifisering av materialer. Denne oppgaven fokuserer på den industrielle anvendelsen sortering basert på klassifisering med hyperspektral avbildning. Sortering av uønskede gjenstander eller defekter kan øke effektiviteten i produksjonslinjer betraktelig.

Denne oppgaven undersøker hvordan en klassifiseringsmodul kan bli laget med et hyperspektralt kamera som sensor og med en ekstern oscillator for synkronisering gjennom triggerpulser. Kameraet som brukes er et kortbølget infrarødt (SWIR) hyperspektralt kamera som skanner linje for linje. Det har spektralområde 930-2500nm og er produsert av selskapet HySpex. Ytelsen og stabiliteten til kameraet blir bekreftet med en gjennomgang av testrapporten. Oscillatoren er en frekvensdeler produsert av selskapet NEOLund, og stabiliteten er bekreftet med et oscilloskop. Klassifiseringsmodulen er laget for å være integrerbar med ethvert sorteringssystem, for eksempel transportbånd med trykkluftventiler eller steg med robotarmer. Denne oppgaven introduserer klassifiseringsmodulen i tre deler: bildeinnsamling og modellering, implementering av en generell klassifiseringsmodul og tilsutt testing med en spesifikk integrasjon.

Objekter beveger seg i forhold til kameraet og blir skannet linje for linje under innhenting av bilder. I denne oppgaven blir seks objekter skannet: tre trebiter og tre steinbiter. Objekter ble valgt for å være enkle å klassifisere for menneskeøyet, slik at resultatene kunne verifiseres manuelt. To klassifikasjonsarbeidsflyter ble opprettet for disse seks objekter. Arbeidsflyt 1 bruker alle de 288 bølgelengdene kameraet kan levere, mens arbeidsflyt 2 begrenser antall bølgelengder til 56. Denne begrensningen utføres basert på informasjon samlet inn under opprettelsen av arbeidsflyt 1. Separasjonen av objekter fra bakgrunnen ble vellykket for arbeidsflyt 1, og klassifiseringsmodellen er nøyaktig. Separasjonsmodellen i arbeidsflyt 2 har mer støy enn arbeidsflyt 1, men er fremdeles vellykket. Klassifiseringsmodellen er enda mer nøyaktig enn den i arbeidsflyt 1.

Den foreslåtte generelle klassifiseringsmodulen er koordinert basert på å samle et fast antall linjer i bilder som er enklere å behandle. Denne samlingen gjør det mulig å sortere objekter basert på førstemann til mølla. En programvare kalt Breeze Runtime fra Prediktera brukes til separasjon og klassifisering av objekter i sanntid. Et program som heter SynchSoftware ble utviklet for denne oppgaven. Den har et brukergrensesnitt som gir brukeren kontroll over Breeze Runtime og frekvensdeleren. SynchSoftware mottar ferdig klassifiserte objekter fra Breeze Runtime, venter til alle objektene som er kortere enn en maksimal størrelse har ankommet, og samler dem inn i bilder.

En test av en spesifikk integrasjon ble utført med en FPGA til å representere sorteringsmaskinen. Frekvensdeleren styrer synkroniseringen ved å sende triggerpulser til kameraet og pulser med 1/128 av frekvensen til FPGAen. FPGAen oppretter deretter tidsstempler basert på pulsene og sender dem til SynchSoftware. SynchSoftware koordinerer disse tidsstemplene hvert sitt bilde og sender dem som byte-strengmeldinger over TCP/IP til FPGA. Synkroniseringen av systemet er avhengig av stabiliteten til frekvensdeleren. Det ble funnet ut at triggersignalet hadde et avvik på mindre enn en promille av perioden. SynchSoftware ble funnet ut til å være begrenset til en kamerafrekvens på under 125Hz. Denne begrensningen skyldes at SynchSoftware bruker mye av tiden på logging. Klassifiseringsmodulen, som består av kameraet, Breeze Runtime, frekvensdeleren og SynchSoftware kan integreres med hvilken som helst treddjeparts sorteringsmaskin.

Acknowledgement

I would like to thank my supervisors, Ian, Julio and Dag Roar, who provided valuable guidance and insight about planning and academic writing. Thank you to my fellow students Sigurd, PG and Pauline for reading through my thesis at its worst and providing valuable feedback. Thank you to the production team at HySpex, especially Martin, Eirik and Ivar, for increasing my understanding of the camera and its calibration. Thank you to Rune for helping me with transporting equipment to and from Poland, and to Antoine for helping with anything.

I would like to thank the "digital study room" consisting of my fellow master candidates. The study room allowed me to feel pressured to work, as well as giving excellent breaks. I would like to mention Carl, Embla, Trym, Halvor, and last and least Kaja for helping me towards the finishing line.

I would like to thank Morten, as well as my sister Alida and brother Mikal for providing housing for me during my stays in Oslo. I want to thank my mom and dad for welcoming me home and giving me the best workspace when we all had to seek refuge at Hovden.

Contents

List of Figures

Symbolslist

A number of components in a PC or PLS model[4]. 12

a index of components (model dimensions, range [1,...,A]). 12

$A_{aperture}$ Entrance aperture area. 5, 9

B matrix of regression coefficients of all **Y**'s[4], size K× M 14

BG_{ref} BackGround reference represents the dark current, i.e. the camera output when there is no incoming light. 9

$\mathbf{b_m}$ regression coefficient vector of the mth y[4], size K× 1 14

C **Y**-weight matrix, the columns are $\mathbf{c_a}$, size: $M \times A$. 14

$\mathbf{c_a}$ **Y**-weight vector of a PLS model, size: $M \times 1$. 14

c Speed of light. 9

DN Raw data output from the camera. 9

E the matrix containing the error and noise (residual information) for **X** in a Principal Component Analysis (PCA) or Partial Least Square (PLS) model, size: $N \times K$. 12–14

F matrix containing the error and noise (residual information) for **Y** in a PLS model, size: $N \times M$. 14

h Planck's constant. 9

i spatial index of the camera pixels (i=1,2,...,I). 9

T_{int} Integration time, the period that the detector captures light for each image taken 9

K number of **X** variables, corresponds to the number of wavelengths. 10

k spectral index of **X** variables (k=1,2,...,K)[4]. 9

L incoming radiance to the camera. 9, 10

λ Wavelength. 9

$\Delta\lambda$ Spectral sampling of camera. 9

M number of $\mathbf{Y}$ variables, corresponds to the number of groups in a PLS-DA model. 14

m index of $\mathbf{Y}$ variables (m=1,2,...,M)[4]. 14

N number of observations. 10

n index of observations (range: $[1, ..., N]$). 12

N_p the number of incoming photons at a pixel during integration time . 9

Ω The solid angle of a pixel. 7, 9

$\mathbf{P}$ the $\mathbf{X}$-loading matrix, the columns are $\mathbf{p_a}$, size: $K \times A$. 12–14

$\mathbf{p_a}$ the $\mathbf{X}$-loading vector of a PCA or PLS model, size: $K \times 1$. 12, 14

p Number of spatial pixels.. 8

T_e The period of the event trigger signal. 28, 29

QE Quantum Efficiency of each spectral band in a photo sensor. 7, 19

R_{norm} Normalized reflectance is the incoming radiance L divided by a reference. The characteristics of the incoming light to the object is therefore removed. 10, 21, 35

RE Relative Efficiency of each pixel in a photo sensor. 7, 18

SF Scaling Factor giving the relationship between the raw camera data output and the photo-electron count. 7

$\mathbf{T}$ the $\mathbf{X}$-scores matrix, the columns are t_a, size: $N \times A$. 12–14

$\mathbf{t_a}$ the $\mathbf{X}$-scores vector of a PCA or PLS model, size: $N \times 1$. 12, 14

T_t The period of the trigger signal. 27–29

θ The angle representing the FOV of the camera. 5

$'$ transpose. 12

v The configured speed of the conveyor belt.[$\mathrm{m\,s^{-1}}$]. 8, 18, 27, 28, 57

wd The distance between the front of the camera and the target surface.[m] 5, 21, 61

W_{ref} Represents the radiance from the white reference. 10

$\mathbf{X}$ matrix of predictor variables, size $N \times K$. 10–15

$\mathbf{Y}$ matrix of response variables, size $N \times M$. 10, 11, 14, 15

$\mathbf{y_m}$ mth vector in $\mathbf{Y}$, size $N \times 1$. 14

Acronyms

CCD	Charge-Coupled Device. 7
FOV	Field Of View. 5, 8, 61, 67, 70
FPGA	Field-Programmable Gate Array. 2, 29, 56, 57, 65, 68
FWHM	Full Width at Half Maximum. 7, 18, 33, 61
GUI	Graphic User Interface. vii, 2, 22, 26, 66
JSON	JavaScript Object Notation. 22, 57
LPS	Lines Per Seconds. 7, 8, 16, 18, 24, 29, 34, 43, 44, 54, 57–60, 66, 67, 70
LSF	Line Spread Function. 7
MCT	Mercury Cadmium Telluride. 7
PC	Principal Component. 12, 13
PCA	Principal Component Analysis. ix, 10, 12–14, 21, 22, 36, 38, 40, 43, 48, 49, 51, 53, 61, 62, 64
PLS	Partial Least Square. ix, 10–12, 14, 15
PLS-DA	Partial Least Square - Discriminant Analysis. 11, 15, 21, 22, 34, 40, 42, 43, 45, 51, 53, 61, 63–65, 70
PSF	Point Spread Function. 7, 18
SNR	Signal to Noise Ratio. 12, 14, 61
SWIR	Short-Wave Infrared. 4, 8, 29, 30, 55, 62

Glossary

across track — parallel to the spatial line of the camera, the x-direction. vii, 5, 6, 18, 21, 63, 67

along track — perpendicular to the spatial line of the camera, also called scan direction and the y-direction. 67

bad pixels — These are pixels that are measured to be too noisy, not responsive enough, have a high dark current, or any other problem that would impede the measurement. These can be found and interpolated away during operation. 7

C# — A programming language that run on the open source .Net Core platform from Microsoft.[5]. 16

correlation — Describes the normalized joint variability of two sets, it is calculated with:

$$Corr(X, Y) = \frac{\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]}{\sigma_{\mathbf{X}}\sigma_{\mathbf{Y}}}$$

11

dynamic range — the relationship between the brightest and darkest light the detector can represent with digital values. 7

frame — A collection of consecutive lines 25

frame size — The number of lines in a frame. 25–28, 57

integrating sphere — A hollow spherical cavity with uniform scattering and narrow entrance and exit apertures. 18, 19

max object size — The maximum length objects can have along track on the sample mover measured in number of lines. Samples larger than this might be omitted. 26–28, 34, 57, 68

overfitting	Overfitting is the use of models or procedures that violate parsimony, that is, that include more terms than are necessary or use more complicated approaches than are necessary.[6] 12, 14
pixel length	the span that each pixel covers along track. 8
pixel width	the span that each pixel covers across track. 8
second order suppression	the second order of a short wavelength will be diffracted in the same direction as the first order of a longer wavelength. Second order suppression is the concept of removing this aliasing effect. 7
Serilog	Serilog[7] is a C# library providing functions to log information to a file as JSON-lines.. 22
spatial resolution	FWHM is used to describe the width of the LSF in the spatial direction y as a measure of optic resolution. 18
spatial smile	a measurement of spectral misregistration. 18, 61
spectral keystone	a measurement of spatial misregistration. 18
spectral resolution	FWHM is used to describe the width of the LSF in the spectral direction λ as a measure of optic resolution. 18
supervised learning	supervised learning use example input-output pairs to make models 14
TTL	A standard used for serial communication, where logical 0 is given by voltages lower than 0.8, and logical 1 is given by voltages higher than 2V. 15, 24, 29
unsupervised learning	unsupervised learning use example input pairs to make models 12
white reference	A material used as a reference to estimate the effects of the light source and ambient light. It should have near perfect diffusive reflectance and known spectral response.. 10
windowing	limitation of the wavelengths the detector records which allows the camera to increase its max framerate.. 7, 34

Chapter 1

Introduction

The thesis suggests a classification module that can be integrated with third-party sorting machines. It is a collaboration with the company HySpex, who produces the camera type used. The suggested classification module is shown as the upper part of figure 1.1. It scans the samples moving across a sample mover, processes those scans in real-time, classifies the objects, collects the objects into frames and sends them to the FPGA. In the figure, an example sorting machine consisting of a vibrating feeder, conveyor belt, light source and pneumatic valves is shown to illustrate one possible configuration. The different parts of the proposed classification module are described below.

Frequency divider

An oscillator that sends trigger pulses and frequency-divided event triggers. The trigger pulses control the camera acquisition, and the camera scans one line for every pulse. The event trigger can be used for coordinating with other systems. The stability of the Frequency Divider signal is tested to verify that it can be used for synchronisation.

Hyperspectral camera

Hyperspectral push-broom (along-track scanner) cameras scan line by line and each spatial pixel in that line has a high spectral resolution. The spectral information can be used to analyse material properties relevant to the classification process. The performance of the hyperspectral camera is documented in a test report by HySpex as part of their quality control process. How these different tests are performed, and the results for the camera are reviewed in this thesis.

Frame grabber

An image transfer protocol used by a frame grabber, an extension card on the computer that communicates with the camera and ensures that every scan line arrives properly. The frame grabber is responsible for numbering the scan lines.

Breeze Runtime

Software from Prediktera[8] made for real-time analysis of hyperspectral data. It used together with Breeze, a software that allows the creation of workflows. These workflows can be executed by Breeze Runtime to do real-time separation of samples from backgrounds and then classifying them. Two classification workflows are made in this thesis. Workflow 1 uses all the camera wavelengths. Workflow 2 limits the wavelengths sent by the camera with windowing. Reducing the number of used wavelengths increases the max speed of the camera and software accordingly.

SynchSoftware

SynchSoftware was developed as part of this thesis to communicate with the third party sorting machines. It is a c# program with a GUI that allows the user to manage the camera, Breeze Runtime and Frequency Divider settings. The software also receives objects from Breeze Runtime, collects them into frames and sorts them. The frames are then sent to the third party sorting machine.

Example sorting machine

A sorting machine is a mechanical system that can separate objects. An example sorting machine is shown in figure 1.1 with the parts listed below. This sorting system can separate a continuous flow of objects into two separate ducts.

- **FPGA:** an Field-Programmable Gate Array (FPGA) that controls the sorting machine.
- **Vibrating feeder:** the shaft the objects enter through, they are distributed evenly with shaking.
- **Conveyor belt:** an endless loop of belt that moves the objects with a constant speed.
- **Pneumatic valves:** a row of valves using air pressure to blow objects into the reject outlet on command.
- **Reject Outlet:** a duct for unwanted objects.
- **Product Outlet:** a duct for wanted objects.

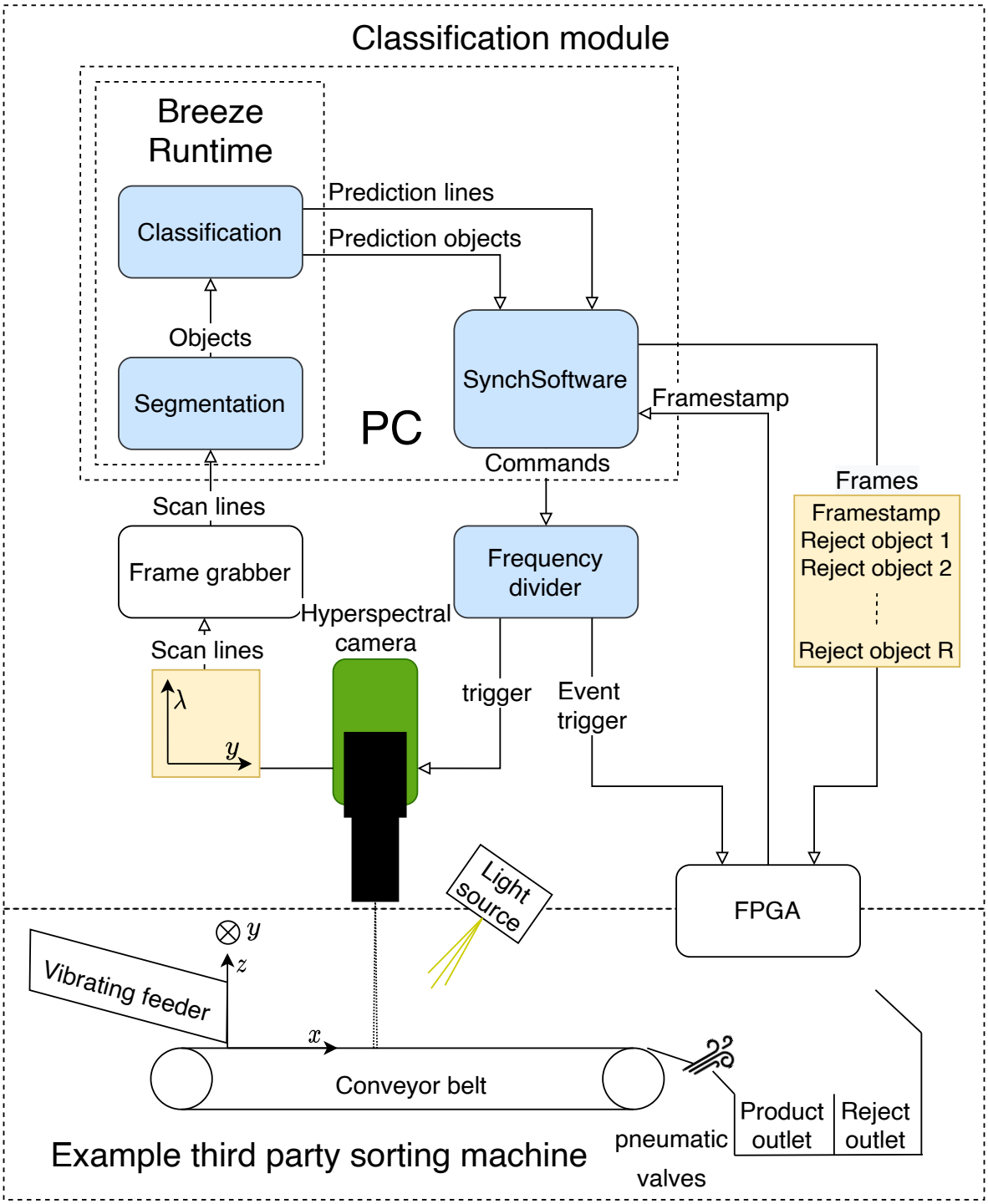


Figure 1.1: Schematic overview of the system. This thesis focuses on the boxes marked with blue, the hyperspectral camera and general integration of all subsystems. The yellow boxes shows the data format at these points.

1.1 Background

Many industries can benefit greatly from having classification modules that can be integrated into any sorting system. Classification based on hyperspectral data and spectroscopy in the visible and near infrared (VNIR), and Short-Wave Infrared (SWIR) domain has been performed for many sample types in many industries:

- Classification based on the quality of Hazelnuts[9]
- Estimating the health of onions[10]
- Detecting defects in Jonagold apples[11]
- Classifying roe, milt and liver from fish[12]
- Material classification of cellulose based materials like pulp, paper and cardboard[13]
- Classification of plastics, papers, glass and metals [14]

The general classification module described in this thesis can utilise the previous research and apply them to industry applications. The mining industry is an example of one such application. In which sorting out unwanted resources at the mine site can increase production capacity, decrease waste disposal and decrease transportation[15].

1.2 Roadmap

The theory section goes through the necessary background to understand the concepts. Small definitions are put in the glossary. Theory has four main sections:

- Hyperspectral imaging: Introduces the necessary theory for understanding how a Hyperspectral camera works and why it can be used to classify samples.
- Multivariate statistics: Goes through the fundamentals of how one can process information containing a huge number of variables. The models PCA and PLS-DA are introduced.
- Hardware: Goes into how two components in the system works. They are responsible for the system staying coordinated.
- Programming: Gives information about important concepts used by SynchSoftware.

Method, results and discussion are all separated into Preparations and Sorting. Preparation focuses on preparing and testing each subsystem to see how they work. That is: characterise the camera, make recordings, create separation and classification models, test the Frequency Divider and test the Camera Link. The Sorting chapter focuses on the communication between the systems and on SynchSoftware.

Method describes how to make recordings and models in a general manner. Results shows the creation of two workflows as well as verification of Frequency Divider and SynchSoftware. The discussion has a test report review, a discussion of the workflows and discussion of the specific integration.

Chapter 2

Theory

This chapter presents the theory and definitions that are the necessary for understanding and discussing the system.

2.1 Hyperspectral imaging

Hyperspectral imaging is also called imaging spectroscopy to reflect that each spatial pixel in the view of a hyperspectral camera is recorded as if seen by a spectrometer. Figure 2.1 shows the progression and transformation of light through a push-broom hyperspectral camera from one spatial pixel at a sample to the spectral row of one spatial pixel on the detector. The light source provides a line of lights that covers the whole Field Of View (FOV). The source of light is shown in figure 1.1. After reflecting off the target, the light travels the working distance(wd) to the close-up lens. The close-up lens is a user-replaceable part made for one specific use case defined by the following parameters:

- wd: The distance between the front of the camera and the target surface.[m]
- θ: The angle representing the FOV of the camera.
- $A_{aperture}$: Entrance aperture area

The close-up lens collimates the light onto the Focusing mirror, which images the scene onto a slit. This slit only allows a narrow line along the y-axis (across track) of the scene to pass through. The Collimating mirror then collimates towards the grating. Both the Focusing mirror and the Collimating mirrors are aspherical mirrors. After collimation, the light comes to the grating, a dispersive element that spreads the light as a function of wavelength. The Lens optics after the grating focuses the light onto the detector, which measures the amount of light that hits its 2D array of pixels during the integration time . The second-order diffraction of a wave with a short wavelength will be diffracted in the same direction as the first order of a wave with a longer wavelength. This effect causes aliasing and must be countered to detect more than one order of wavelengths. A suppression filter mounted on the surface of the detector achieves this for push-broom cameras.

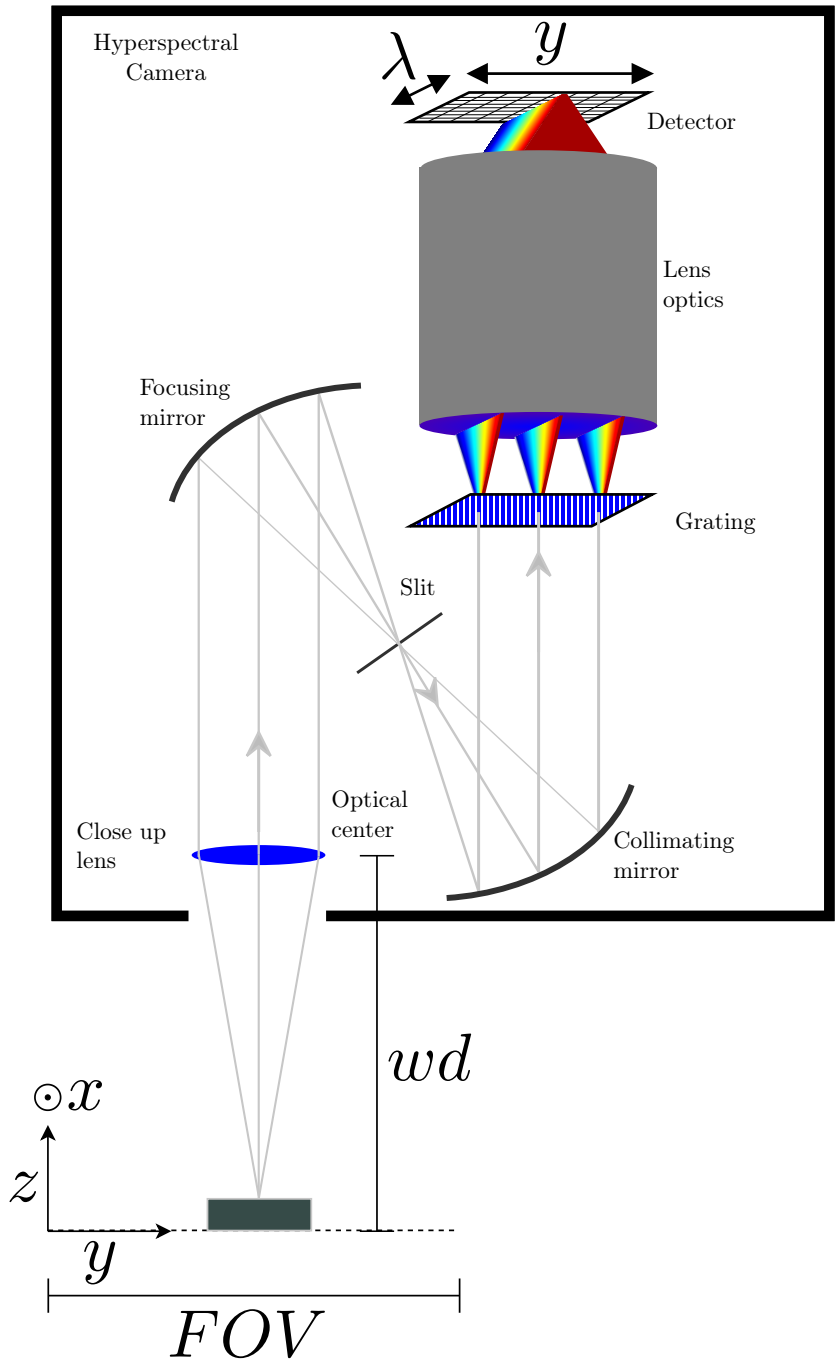


Figure 2.1: across track view of a hyperspectral camera pointing towards the conveyor belt. The figure is inspired from a figure made by[1]

2.1.1 Detector

The variables and acronyms used in this chapter:

- QE: Quantum Efficiency of each spectral band in a photo sensor
- RE: Relative Efficiency of each pixel in a photo sensor
- SF: Scaling Factor giving the relationship between the raw camera data output and the photo-electron count
- Dynamic range: the relationship between the brightest and darkest light the detector can represent with digital values.
- Bad pixels: These are pixels that are measured to be too noisy, not responsive enough, have a high dark current, or any other problem that would impede the measurement. These can be found and interpolated away during operation.
- Second order suppression: the second order of a short wavelength will be diffracted in the same direction as the first order of a longer wavelength. Second order suppression is the concept of removing this aliasing effect.
- Charge-Coupled Device (CCD)
- Mercury Cadmium Telluride (MCT)
- Lines Per Seconds (LPS)
- Windowing: limitation of the wavelengths the detector records which allows the camera to increase its max framerate.
- Ω: The solid angle of a pixel

The detector is a 2D array MCT sensor. It is a global shutter type sensor, which means that all the pixels are read simultaneously, and there are no rolling shutter effects[16]. MCT is a semiconductor material with a band-gap that can be tuned in the range 85.47nm - 25000nm during production[17]. Due to black-body radiation, a detector measuring wavelengths in the infrared region must be cooled down, or else it will spontaneously generate photo-electrons. The Global shutter has a windowing function which allows the user to specify which detector columns to record. For hyperspectral imaging, this function can be used to limit the wavelengths used. Maximum LPS can be increased proportionally to the number of reduced wavelengths.

2.1.2 PSF

Point Spread Function (PSF) describes the way diffracted beam hits a 2D surface, not as a perfect dot but as an airy disk. In reality PSF looks like the aftermath of a rock dropped in water, a middle peak and repeating but diminishing heights in all directions. However, when the optics are optimised for minimising higher-order diffraction (the heights to the sides), PSF can be approximated by a Gaussian distribution.[18]. This Gaussian distribution is described by the peak position and by Full Width at Half Maximum (FWHM). Line Spread Function (LSF) is this same gaussian distribution centred on a line.

2.1.3 Square pixels

This section uses the values and acronyms stated below:

- v: The configured speed of the conveyor belt.[$\mathrm{m\,s^{-1}}$]
- LPS
- Pixel width
- Pixel length
- p: Number of spatial pixels.
- FOV

The definition of a square pixel is given in (2.1).

$$\text{Pixel length} = \text{Pixel width} \tag{2.1}$$

Where pixel width is calculated from the span (FOV) seen by the camera and the number of spatial pixels (p) as seen in (2.2)

$$\text{Pixel width} = \frac{FOV}{p} \tag{2.2}$$

Pixel length is found from v and LPS as seen in (2.3).

$$\text{Pixel length} = \frac{v}{\text{LPS}} \tag{2.3}$$

Then the three equations (2.1), (2.2), (2.3) are combined and solved for LPS (2.4).

$$\begin{aligned} \text{Pixel length} &= \text{Pixel width} \\ \frac{v}{\text{LPS}} &= \frac{FOV}{p} \\ \text{LPS} &= \frac{v \cdot p}{FOV} \end{aligned} \tag{2.4}$$

2.1.4 Short wave infrared

The spectral range 690 nm - 3000 nm interacts with overtone and combination bands of fundamental molecular vibrations, especially stretching and bending, but some deformation as well.[19]

The resultant spectra do not have well-defined peaks that correspond precisely to the molecule vibrations in a solid, but rather a complicated mixture of fundamental vibrations and overtones. This phenomenon means that the SWIR spectral range is not that well suited for quantifying specific components. Instead, this complicated mixture provides a fingerprint of each material composition.

2.1.5 Radiometric calibration

The variables and acronyms used in this chapter:

- i: spatial index of the camera pixels (i=1,2,...,I)
- k: spectral index of $\mathbf{X}$ variables (k=1,2,...,K)[4]
- DN: Raw data output from the camera
- N_p: the number of incoming photons at a pixel during integration time
- BG_{ref}: BackGround reference represents the dark current, i.e. the camera output when there is no incoming light
- T_{int}: Integration time, the period that the detector captures light for each image taken
- $\Delta\lambda$: Spectral sampling of camera
- λ: Wavelength
- L: incoming radiance to the camera
- c: Speed of light
- h: Planck's constant

Incoming light (N_p) to a hyperspectral camera differs from the output (DN) of a sensor. The difference is given by (2.5).

$$DN[i,k] = N_p[i,k] \cdot QE[k] \cdot RE[i,k] \cdot SF + BG[i,k] \tag{2.5}$$

$N_p[i,k]$ is a more useful variable than DN as it is invariant of the detector used and describes how many photons hit each pixel during T_{int}.

$$N_p[i,k] = \frac{L[i,k] \cdot t_{int} \cdot A \cdot \Omega \cdot \Delta\lambda[k] \cdot \lambda[k]}{h \cdot c} \tag{2.6}$$

As can be seen in (2.6), N_p is also dependent on the following camera specific parameters $A_{aperture}$, Ω and $\Delta\lambda$. As well as the use case specific parameters T_{int} and λ.

The parameter $L[i,k]$, is invariant of both sensor and camera properties, i.e. only dependent on incoming radiance. This independence is desirable for getting the same results under any condition with any camera, and to follow SI standards. The two equations (2.5) and (2.6) can be combined and then solved for L to give (2.7). This function is applied to all the scans recorded with the camera.

$$L[i,k] = \frac{(DN[i,k] - BG[i,k]) \cdot h \cdot c}{QE[k] \cdot RE[i,k] \cdot SF \cdot t_{int} \cdot A \cdot \Omega \cdot \Delta\lambda[k] \cdot \lambda[k]} \tag{2.7}$$

2.1.6 Normalized reflectance

Parameters discussed in this chapter:

- R_{norm}: Normalized reflectance is the incoming radiance L divided by a reference. The characteristics of the incoming light to the object is therefore removed.
- W_{ref}: Represents the radiance from the white reference

Section 2.1.5 describes how to find the actual radiance L that is coming in to the camera. One step further is to remove the effects of the light source and ambient light, so that the only effect shown by the recording is the reflectance of the samples and background. The effect of the light source and ambient light can only be removed if the light is characterised. This characterisation is done by recording a white reference with a known spectral response right before recording the samples. The light can then be normalised based on this reference, as seen in (2.8).

$$R_{norm} = \frac{L[i,k]}{W_{ref}i,k - BG[i,k]} \tag{2.8}$$

2.2 Multivariate statistics

The variables that are necessary for discussing multivariate statistics are listed below:

- $\mathbf{X}$: matrix of predictor variables, size $N \times K$
- $\mathbf{Y}$: matrix of response variables, size $N \times M$
- N: number of observations
- K: number of $\mathbf{X}$ variables, corresponds to the number of wavelengths

Statistics with more than one variable analysed simultaneously is called multivariate statistics. For hyperspectral imaging, there are several hundred correlated variables for each pixel. This high amount of variables related to each other makes it necessary to simplify and extract the interesting features. Both for further analysis and to be able to understand the data. As Fredrik Pettersson writes:

"A dataset consisting of N observations and K variables can theoretically be visualized as a plot of N samples in a K dimensional space. However, the limitations of our cognitive abilities make it very difficult to visualise such spaces when $K > 3$."[20, p. 33]

So, to make the data processable for our brains, the multivariate projection methods PCA and PLS will be introduced. The goals of these methods are listed below:[2]:

- PCA
  - Data clean up: identify background, remove noise.
  - Find relationships between observations(samples): trends, groups, outliers.
  - Find relationships between variables: groupings, correlation.

- PLS
  - Find relationships between two blocks of variables ($\mathbf{X}$ and $\mathbf{Y}$).
  - Find models to quantify the contents in samples.
- Partial Least Square - Discriminant Analysis (PLS-DA)
  - Find models to classify new unknown samples.
  - Find models to understand differences between known classes.

These models and methods give an insight into the data, making it easier to find the relationship between factors and responses.

Projection methods

When working with data that contains a high number of dimensions, such as hyperspectral images, it is useful to reduce the number of dimensions. Doing so can make it possible to visualise the critical features in a simple 2D plot, and make further processing of the data more useful and understandable. Data with a high number of dimensions also often have a high correlation between neighbouring dimensions; this is a problem as linear modelling methods usually requires uncorrelated variables. Reducing the number of dimensions can help mitigate this problem.

A projection is a mapping from a high dimensional space onto a lower-dimensional space. Figure 2.2 illustrates this with a three-dimensional space being projected down into a one-dimensional space. Dimensions like that one are called latent variables because they cannot be measured directly, but are still variables that represent the data. In this example, the sample information is compressed from three variables to one. There is naturally data loss in this process, and it is the goal for projection algorithms to keep the beneficial information while removing obsolete and uninteresting data, like noise. In other words: projection algorithms aim to find the underlying structures describing the samples.

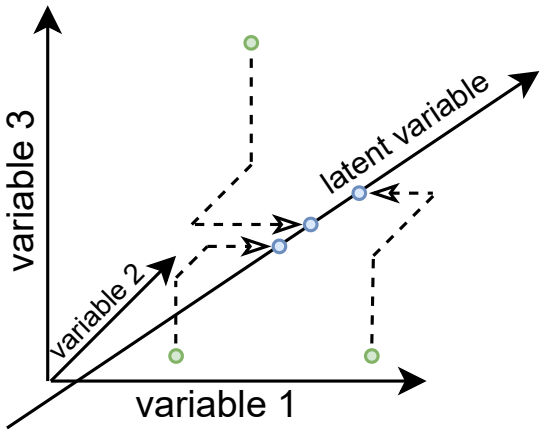


Figure 2.2: Example of projection from three variables to one latent variable.

The latent variables are found by focusing on some characteristic in the data and making dimensions that maximise or minimise that characteristic. The first latent variable is calculated from the original

dataset. The second latent variable is calculated from the remainder of that calculation. This process can be repeated until the desired amount of dimensions has been made. Each latent variable is associated with a percentage value of how much of the dataset they describe. Two projection methods are introduced below, PCA and PLS.

2.2.1 Principal Component Analysis (PCA)

The variables that are necessary for discussing PCA are the ones introduced for multivariate statistics and the ones listed below:

- a: index of components (model dimensions, range [1,...,A])
- A: number of components in a PC or PLS model[4]
- n: index of observations (range: $[1, ..., N]$)
- $\mathbf{t_a}$: the $\mathbf{X}$-scores vector of a PCA or PLS model, size: $N \times 1$
- $\mathbf{T}$: the $\mathbf{X}$-scores matrix, the columns are t_a, size: $N \times A$
- $\mathbf{p_a}$: the $\mathbf{X}$-loading vector of a PCA or PLS model, size: $K \times 1$
- $\mathbf{P}$: the $\mathbf{X}$-loading matrix, the columns are $\mathbf{p_a}$, size: $K \times A$
- $\mathbf{E}$: the matrix containing the error and noise (residual information) for $\mathbf{X}$ in a PCA or PLS model, size: $N \times K$
- $'$: transpose
- Signal to Noise Ratio (SNR)
- Critical distance: How different a measurement can be from the model, and still be included.

PCA is a unsupervised learning method that finds the latent variables that describe the most variance in a dataset. For hyperspectral images that can have hundreds of wavelengths for each pixel, this is a powerful tool for simplification and visualisation of the data. It makes it possible to show the essential parts of the data in a simple 2D plot.

An important decision is to find out how many dimensions that should be used in the model (A). The error matrix $\mathbf{E}$ gets smaller as more dimensions are introduced. Each new dimension increases computational time, and having too many dimensions can lead to overfitting. An overfitted model can only describe the training data accurately and is not able to describe new data. PCA needs a large SNR to function properly. If the noise is too large compared to the signal, the model will describe structures in the noise.

The dimensions of a PCA model are called Principal Components (PCs). They each consists of two vectors, the loading vector $\mathbf{p_a}$ and the score vector $\mathbf{t_a}$. $\mathbf{p_a}$ is a vector that describes the variation in the variable direction, with one value for each variable. Variables that have similar values in $\mathbf{p_a}$ exhibits similar properties across the observations. $\mathbf{t_a}$ contains one value for each observation in $\mathbf{X}$ telling where along the dimension spanned by $\mathbf{p_a}$ that observation lands. These values describe the variation in the sample direction and can be interpreted to find differences and similarities between samples.

Figure 2.3 and equation (2.9) shows how these values corresponds to each other. $\mathbf{X}$ is decomposed into the two matrixes $\mathbf{T}$ and $\mathbf{P}$, which if you multiply together forms an approximation of $\mathbf{X}$. The error of this approximation is stored in the residual matrix $\mathbf{E}$.

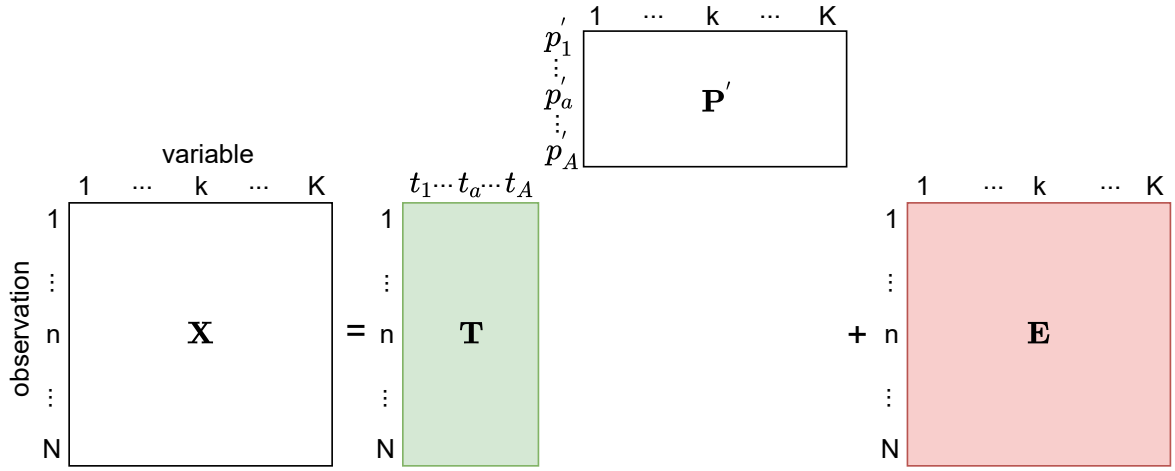


Figure 2.3: An illustration of (2.9). Reducing the dimensions from K to A introduces an error. The better the model the smaller the values in $\mathbf{E}$. This illustration is based on [2, p. 82].

$$\mathbf{X} = \mathbf{T}\mathbf{P}^{'} + \mathbf{E} \tag{2.9}$$

The first PC(PC1) is created by finding the latent variable that has the least sum of square distances to the observations. The latent variable will then have the minimum possible variance from itself to the observations. This is equivalent to finding the latent variable that describes the most variance in the dataset.

The second PC(PC2) is found in the same manner, but with the restriction that it has to be orthogonal to PC1. This is achieved by finding PC2 from $\mathbf{E}_1$, the residual from PC1. PC3 has to be orthogonal to both PC1 and PC2, so it is found from $\mathbf{E}_2$, the residual from PC2. This can go on until the number of PCs is equal to the number of observations or the number of variables, whichever is the smallest.

The advantages of PCA are the following, adopted from [21, p. 38]:

- Simplification
- Data reduction
- Modeling
- Outlier detection
- Variable selection
- Classification
- Prediction
- Unmixing

These advantages have made PCA a fundamental tool in processing of hyperspectral images.

2.2.2 Partial Least Square

The variables that are necessary for discussing PLS are the ones introduced for multivariate statistics and PCA, and those listed below:

- $\mathbf{y_m}$: mth vector in $\mathbf{Y}$, size $N \times 1$
- M: number of $\mathbf{Y}$ variables, corresponds to the number of groups in a PLS-DA model
- m: index of $\mathbf{Y}$ variables (m=1,2,...,M)[4]
- $\mathbf{b_m}$: regression coefficient vector of the mth y[4], size K$\times$ 1
- $\mathbf{B}$: matrix of regression coefficients of all $\mathbf{Y}$'s[4], size K$\times$ M
- $\mathbf{c_a}$: $\mathbf{Y}$-weight vector of a PLS model, size: $M \times 1$
- $\mathbf{C}$: $\mathbf{Y}$-weight matrix, the columns are $\mathbf{c_a}$, size: $M \times A$
- $\mathbf{F}$: matrix containing the error and noise (residual information) for $\mathbf{Y}$ in a PLS model, size: $N \times M$

PLS is a supervised learning method for making models of data with a high number of variables. Both the ingroup variance in $\mathbf{X}$ and $\mathbf{Y}$, and the covariance between the two, is used to find out if $\mathbf{X}$ contain information about $\mathbf{Y}$.

Advantages of PLS:

- Can have a high variable to observation ratio.
- Can handle high correlation between variables.
- Decreases computational load.

Disadvantages of PLS:

- Chance of overfitting.
- Needs manual labeling for training.
- Needs a high SNR.

PLS can analyse data with strongly collinear (correlated), noisy, and numerous $\mathbf{X}$-variables, and also simultaneously model several response variables, $\mathbf{Y}$.

Model $\mathbf{X}$ as:

$$\mathbf{X} = \mathbf{t_1 p_1}^{'} + \mathbf{t_2 p_2}^{'} + ... + \mathbf{t_a p_a}^{'} + \mathbf{E} = \mathbf{T P}^{'} + \mathbf{E} \tag{2.10}$$

Model $\mathbf{Y}$ as:

$$\mathbf{Y} = \mathbf{t_1 c_1}^{'} + \mathbf{t_2 c_2}^{'} + ... + \mathbf{t_a c_a}^{'} + \mathbf{F} = \mathbf{T C}^{'} + \mathbf{F} \tag{2.11}$$

When these models have been made, linear regression can be performed on the new dimensions. The quality of this linear regression can be evaluated from the two parameters, R^2 and $RMSE$:

R^2 gives an upper bound of how well $\mathbf{Y}$ is predicted from $\mathbf{X}$ by the model[4]. The closer to 1 the better.

$RMSE$: Equation 2.12 calculates root mean square error for a prediction and can be used to determine how accurate the prediction is on a labelled dataset.[22]

$$RMSE = \sqrt{\frac{\sum_{i-1}^{n}(E[y_i] - y_i)^2}{n}} \tag{2.12}$$

2.2.3 Partial Least Square - Discriminant Analysis

PLS-DA uses the same philosophy as PLS, but instead of having continuous variables in the training set, only the values 0 and 1 is used:

"A PLS-DA model can be seen as a regular PLS model with binary and discrete responses instead of continuous response.[20, p. 37]"

Figure 2.4 shows an illustration of how these values are set up.

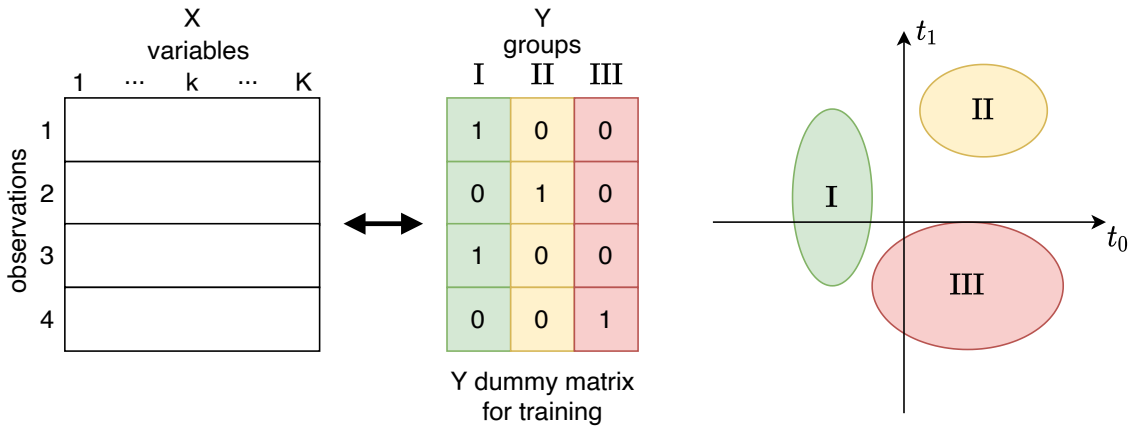


Figure 2.4: An example of how a dummy matrix is used for labeling in PLS-DA, based on a figure from [3]

2.3 Frequency Divider

The frequency divider is a custom made card designed at NEOLund. The main component of the frequency divider is the microcontroller ATMEGA8-AI. It is used to process all inputs and to generate the outputs signals. The chip can be interfaced through serial commands sent over TTL. It can provide a pulse that can be used to trigger camera acquisition. It can also provide a pulse that is frequency divided by any power of two between 2 and 256.

2.4 Camera Link

Camera Link is a protocol used for video transfer in industry applications, and it allows for a fast and reliable transfer of image data. Each line scanned by the camera is sent as an image to the camera link. The Camera Link ensures that all the images are received and that they are in the correct order.

The Camera Link creates a timestamp the moment an image from the camera is loaded into memory. This timestamp could be used to synchronise the images with the pneumatic valves, but only if they are stable enough. The timestamps sent from the Camera Link was investigated to check if they are stable and accurate enough to use for synchronisation.

2.5 Programming

2.5.1 BreezeClientCore

BreezeClientCore is a small library written by Prediktera in C# to ease implementation with Breeze Runtime. It has all the necessary functions for communicating with and configuring the camera and Breeze Runtime.

Listening to predictionLines

BreezeClientCore has functionality for listening to DataStreams and ServerEvents from Breeze Runtime. This is utilised by SynchSoftware. Some of the implementation is introduced here for later discussion.

The thread listening to the predictionLines does not sleep, and the predictionLines are read concurrently as can be seen from the code in listing 2.1. It says ReadAsync, which implies that the stream is read asynchronously in another thread, but the next command says to wait for the task to finish reading. This means that it waits for the current line to be finished before looking for the next one. It looks for a line until the timeout of 250ms. The timeout will only be activated if the cameras LPS is set to below 4Hz, which there is no reason to do, or if there is a transmission error.

```
616     ...
617     while (totalReadByteCount < count)
618     {
619         var task = stream.ReadAsync(buffer, 0, count - totalReadByteCount,
                cancellationToken);
620         task.Wait(cancellationToken);
621         ...
```

Listing 2.1: BreezeClientCore code. The thread listens to predictionLines continuously.

Listening for events

The code in listing 2.2 shows a snippet from this code describing how the thread will wait for 100ms if there is no data available. When the data has been read an event will be called which I listen to.

```
423     ...
424     while (!_stopListeningToServerEvents)
425     {
426         if (!stream.DataAvailable)
427         {
428             Thread.Sleep(100);
429             continue;
430         }
431         var json = streamReader.ReadLine();
432         ...
```

Listing 2.2: BreezeClientCore Code. The thread sleeps for 100ms when waiting for objects and errors (ServerEvents)

Chapter 3

Method

3.1 Preparations

Lab setup

Figure 3.1 shows how the camera is mounted when recording. It points across track on a sample mover, with a light source illuminating the samples from the right. A white reference is placed to the left of the samples and is recorded before recording samples. The speed v of the sample mover and the acquisition rate LPS of the camera must be coordinated to get square pixels in the recordings as described in section 2.1.3.

Camera characterisation

Calibrations and characterisations are performed as part of the camera production. Calibration is necessary for any sensor to give a reliable and accurate representation of what it measures. A test report is created for each HySpex camera, describing which tests it has gone through and the results. The test report is included in the appendix of this thesis. The following list describes how the tests are performed:

1. Spatial resolution: Measured by pointing the camera at a point source with a broad spectre and recording FWHM. The point source is simulated to be infinitely far away with a collimator.

2. Spectral keystone: Uses the same point source configuration as above, but measures the centre of gravity of the PSF.

3. Spectral resolution: Measured by illuminating an integrating sphere with different light sources with very narrow bands and recording FWHM.

4. Spatial smile: Uses the same lights as above, but measures the centre of gravity of the PSF.

5. Responsivity matrix: RE is found by pointing the camera towards an integrating sphere with one known white light source with a broad spectral curve.

6. Quantum efficiency matrix: QE is found by pointing the camera towards an integrating sphere with known light sources inside that have very narrow peaks at specific wavelengths. These light sources are turned on in sequence. Several light sources with different peaks across the whole spectrum of the sensor should be used, and then the intermediate pixels can be interpolated.

7. Second-order suppression: Tested by pointing the camera at an integrating sphere where an optic low-pass filter is placed at the aperture. There should not be any response from the wavelengths longer than the cut off of the low-pass filter.

8. Signal/noise: Estimated by pointing the camera into an integrating sphere illuminated by a stable DC-driven non-flicker broadband illumination.

9. Bad pixels: Any pixel that is found to not be responsive enough or too responsive during the other tests are saved as a bad pixel to be interpolated away.

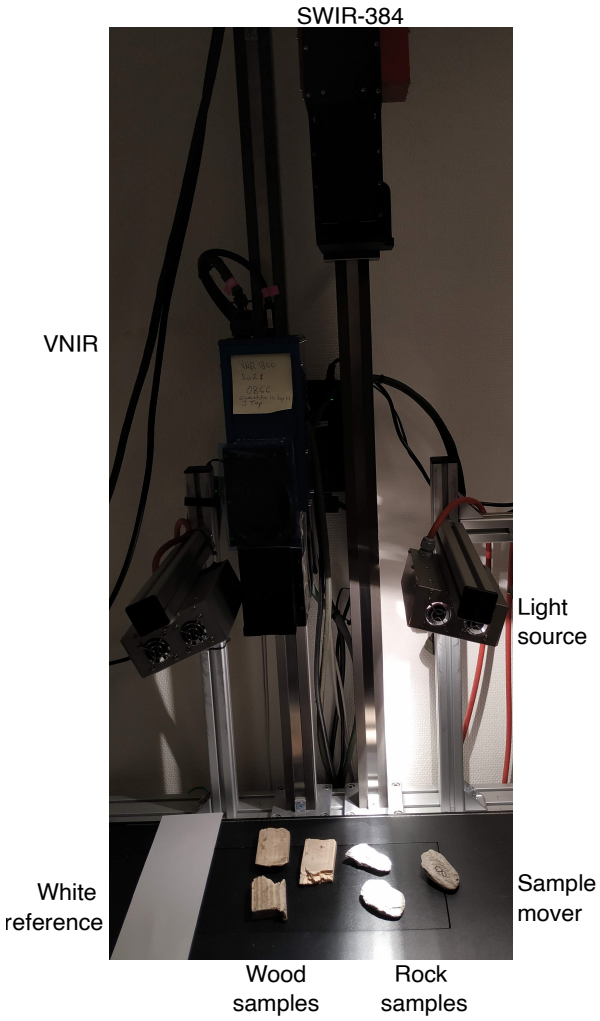


Figure 3.1: Picture of the lab setup, only the SWIR-384 was used in this thesis, not the blue VNIR camera. Three wood and three rock samples are used.

3.2 Image acquisition

To be able to make a model, data is necessary. The purpose of this step is to gather the data that is later used to make PCA and PLS-DA models. The Hyperspectral camera needs to be mounted so that it points perpendicular(across track) to the moving direction of the translation stage. The camera needs to have a lens attached, and be mounted with the correct wd for that lens. The camera and translation stage is connected to the computer and controlled by Breeze. The light source should have its focus on the line seen by the camera. A white reference should be recorded under the same conditions as the samples. This recording will later be used to find R_{norm}.

The samples to be recorded should cover as much of the variation within the materials as possible. The steps of doing a recording are listed below:

- Turn on the lights, place the samples and the white reference at two different spots on the translation stage.
- Point the camera at the white reference and check the integration time. The saturation should be around 80-90%. If some of the samples are more reflective than the white reference, then it is a good idea to check the saturation while pointing to them. This step is to ensure that the camera captures a good colour depth without going into max saturation.
- Choose the number of frames to take for white and dark reference. These frames will be averaged, and a higher number might make a more accurate reference.
- Check the focus of the camera, which is done by maximising the contrast seen by the camera. Adjust the camera up and down until maximum contrast is achieved.
- Take a white reference. It is important to move the stage while taking the reference. The camera will find structures in the white reference if it points towards the same spot throughout the recording.
- Take a dark reference with the camera shutter closed.
- Record the samples group by group.

3.3 Modelling

A software suite called Breeze, developed by the company Prediktera in Sweden, is used to record and analyse the hyperspectral data from the camera. Two programs from the suite are used; Breeze and Breeze Runtime. Breeze is used for recording and making models and Breeze Runtime is used for real time applications. Breeze uses PCA to do the separation between samples and background. This is a much explored technique [20] [23] [24] [21]. Breeze uses PLS-DA for the classification. This is also a widely explored method: [25] [4] [20]. The thesis shows that these methods can be applied directly to sorting scenarios.

3.3.1 Separation with PCA

Before samples can be classified, they need to be localised. The method PCA that is described in section 2.2.1 is well suited for this. Both because it is an unsupervised method and because it can

work on data with a high dimensionality and small sample size. An unsupervised method fits well here because a training set for supervised learning would need to be labelled pixel by pixel. By using PCA first, the samples can be identified and then be labelled sample by sample for classification.

PCA was originally designed to be an exploratory technique. This technique let one see the combinations of dimensions that give the highest and second-highest amount of variance. They are plotted against each other in a 2D scatter plot. This plot will often reveal interesting information that tells much about the properties of the materials in the image. Breeze is a program that facilitates this exploration, and that can make a model that separates the pixels corresponding to samples from the ones corresponding to background. In Breeze parts of the scatter plot can be selected to show what part of the "real" image they correspond with. Parts of the scatter plot can be removed or selected to create a new scatter plot. The goal is to create a scatter plot that describes the samples, but not the background. The results of this exploration can be used to separate similar samples from the background. The last part of making a model is choosing the critical distance. Critical distance is a number that describes how far from the model a sample can be, and still be included in the model.

3.3.2 Classification with PLS-DA

The classification model made in Breeze uses PLS-DA to reduce the number of dimensions and then performs linear regression on the new dimensions. PLS-DA is a statistical method that finds the combination of dimensions that describe the highest covariance between, and lowest variance within, each material group. The first step of making a classification model is to manually label the samples that were localised with PCA. Breeze can then make a PLS-DA model based on which samples corresponds to which group. While making a PLS-DA model, the wavelengths that do the most for the classification can be identified, this will be utilised when making workflow 2.

3.4 General sorting system

The systems discussed in this section:

- Frequency divider - Sends the trigger and event trigger pulse.
- Hyperspectral camera - Gather the spectral data.
- Breeze Runtime - Performs the separation and classification.
- FPGA - The example brain of a third party sorting machine.

The sorting system is the combination of these four. This section is going to describe how to set it up and use the first three. The main coordinator is SynchSoftware, a software written by the author that initialises and coordinates the different parts of the system. SynchSoftware is a c# program with a WPF[26] GUI interface. It is made for the purpose of being a connection between Breeze Runtime and a third party sorter. It also manages the settings, initialises, and starts and stops the four systems listed in the start of the section. SynchSoftware logs everything it does and all the information it receives as JavaScript Object Notation (JSON)-lines through the library Serilog. An array of counters in the SynchSoftware GUI allows the user to monitor the status of the sorting in real time.

3.4.1 Communication

Figure 3.2 shows the communication pathways in the system. SynchSoftware communicates with three systems: Breeze Runtime, the Frequency Divider and a third party sorting machine. It tells the Frequency Divider when to start sending trigger pulses and with what period, as well as how often the event trigger pulse should be sent compared to the trigger pulse. SynchSoftware is used to choose workflow for Breeze Runtime and what type of sample to reject. It is used to start the Hyperspectral Camera acquisition and the Framestamp stream from the FPGA. SynchSoftware receives the output from Breeze Runtime, collect it in frames and sends it to the FPGA. The errors from all systems are received and logged by SynchSoftware.

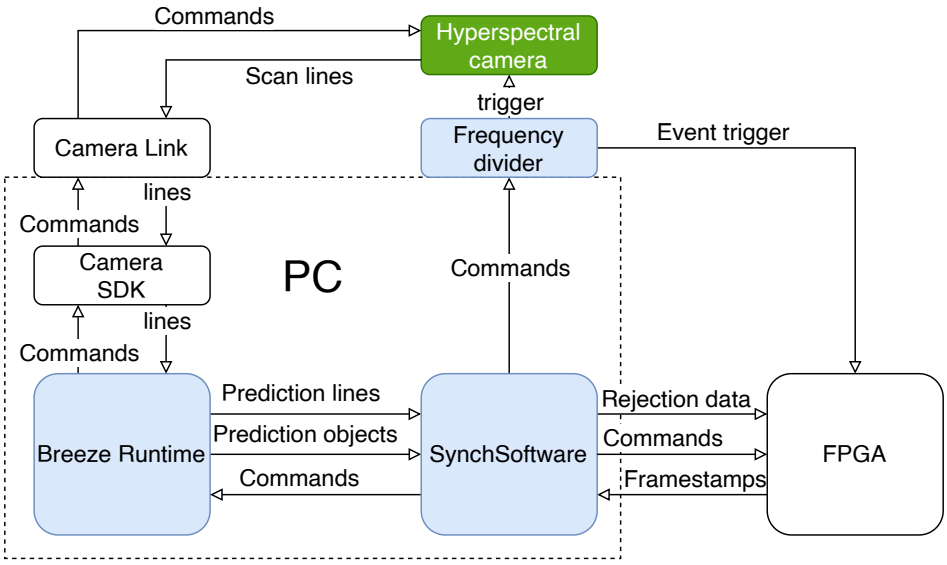


Figure 3.2: The circle communication of the system. SynchSoftware is the central control of the system.

Breeze Runtime

Breeze Runtime is the software used to identify and classify the samples in the scans.

Communication with Breeze Runtime is over TCP/IP, through unindented JSON ended with the symbols CR + LF (ASCII 13 + ASCII 10)[27]. This communication is handled through BreezeClientCore, introduced in section 2.5.1

SynchSoftware has three TCP/IP connections with Breeze Runtime:

- Commands: the connection used for sending camera and breeze runtime commands and read the responses. Both camera and Breeze Runtime settings are changed through these commands.
- Event stream: the connection where Errors, Events and classified objects get sent from Breeze Runtime.
- Datastream: the connection where Line Prediction gets sent from Breeze Runtime.

Frequency divider

The communication with the Frequency Divider is done over serial port with the TTL protocol. The communication is initialised as in code listing 3.1. The function OnFreqDividerDataReceived listens to the DataReceived event and handles the incoming stream of information by writing it to the DebugMessage window in SynchSoftware and logging it. Data is sent to the Frequency Divider with the SerialPort function WriteLine: `FreqDivider.WriteLine("command");`.

```
92  FreqDivider = new SerialPort
93  {
94      BaudRate = 250000,          //bits per second
95      Parity = Parity.None,
96      DataBits = 8,               //8 bits per byte
97      StopBits = StopBits.One,
98      Handshake = Handshake.None,
99      ReadTimeout = 500,          //ms
100     WriteTimeout = 500          //ms
101 };
102 FreqDivider.DataReceived += OnFreqDividerDataReceived;
```

Listing 3.1: SynchSoftware code. Initialisation of serial port for the Frequency Divider

The frequency divider controls the timing and needs to be set up for the correct LPS. This is done by sending a serial command with the period. The frequency divider will then send a trigger pulse to the camera with the set period, and also an event trigger to the FPGA. The event trigger can be frequency divided by a power of 2 between 2 and 256.

3.4.2 SynchSoftware GUI

SynchSoftware can be used to start the following actions:

- Breeze Runtime and camera functions:
  - Choose workflow
  - Choose class to reject
  - Use simulator camera to simulate a connection.
  - Take dark reference
  - Take white reference
  - Connect to camera
  - Load camera info
  - Start prediction
  - Stop prediction
  - Start recording
  - Stop recording
- Frequency divider:

  - Set period between trigger pulses.
  - Set the frequency division of the event trigger.
  - Start the trigger pulses.
  - Stop the trigger pulses.
  - Send any serial command.

- Third party sorting machine:
  - Set IP address and ports for connection.
  - Set frame size: How many lines to collect for each frame.
  - Set max object size: the max size of objects before they get sorted wrong.
  - Set timeout: How long to wait when connecting.

Figure 3.3 is a screenshot of SynchSoftware.

3.4.3 SynchSoftware internal logic

Listing 3.2 shows how the linenumber is added to a dictionary with the ticks timestamp as a the key.

```
402 ...
403 predictionLineDictionary.Add(predictionLine[0].CaptureTime.Ticks,
        predictionLine.FrameNumber);
404 ...
```

Listing 3.2: SynchSoftware code. The code for adding to predictionLineDictionary

Listing 3.3 shows the code used to find linenumbers for the top and bottom of samples.

```
534 ...
535 int pixelTop = predictionLineDictionary.GetFrameNumber(predictionObject.
        CaptureTimeFirstLine.Ticks)
536 int pixelBottom = predictionLineDictionary.GetFrameNumber(predictionObject.
        CaptureTimeLastLine.Ticks)
537 ...
```

Listing 3.3: SynchSoftware code. Looking up linenumber in the dictionary

3.4.4 SynchSoftware frame collection

New values used in this section:

- Frame: A collection of consecutive lines
- Frame size: The number of lines in a frame

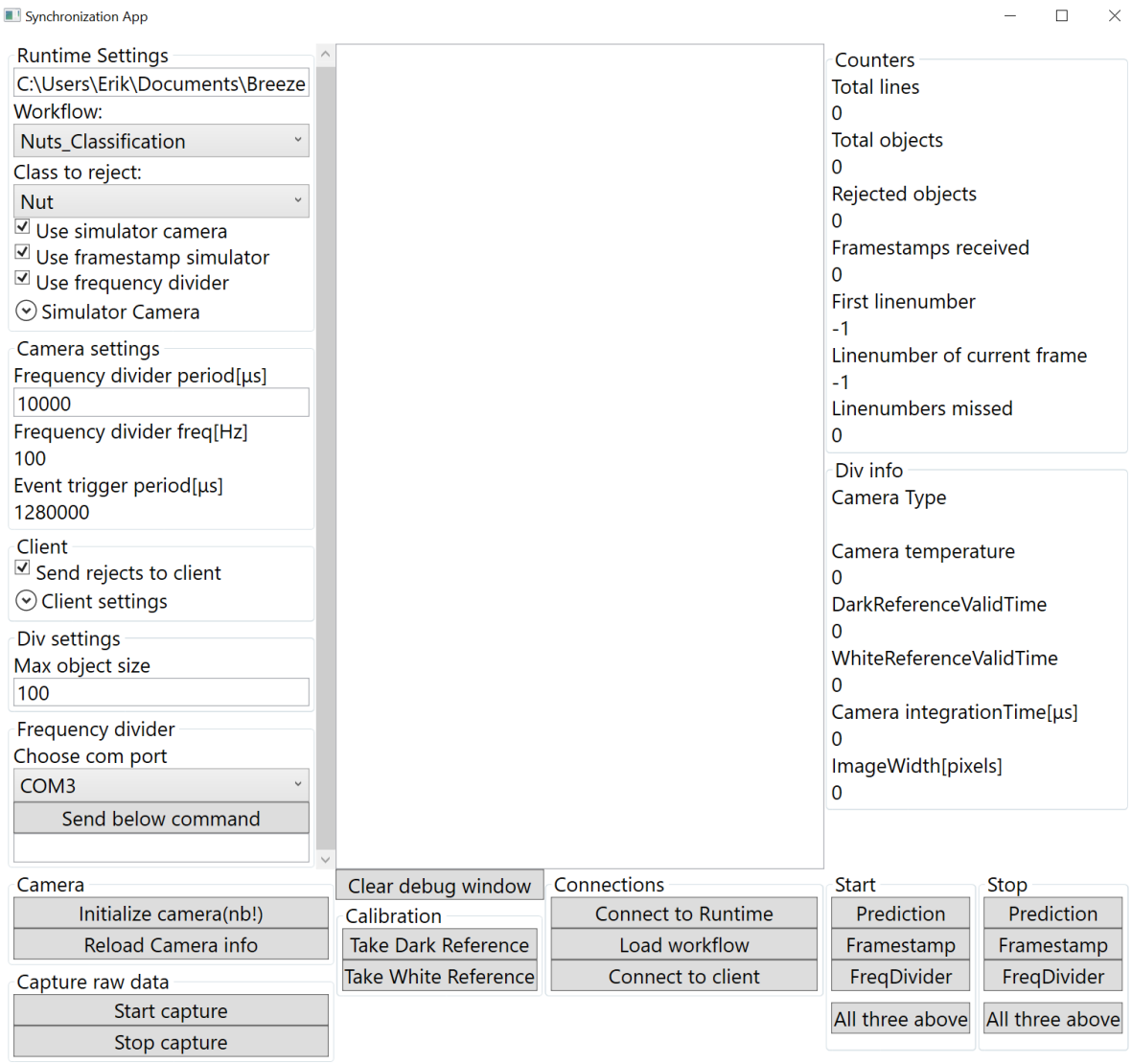


Figure 3.3: The GUI of SynchSoftware.

- Max object size: The maximum length objects can have along track on the sample mover measured in number of lines. Samples larger than this might be omitted.

The objects should come in an order defined by the first line of the object. Breeze Runtime sends the objects as soon as they have finished classifying them. This effectively means that the objects are sorted in an order defined by the last line of the object. To be able to sort the objects, SynchSoftware must save up objects coming from a certain number of lines. This collection of lines and their objects will be called a frame. This makes it necessary with two new parameters: frame size and max object size. frame size is the number of lines that are collected together to form one frame. max object size is the number of lines that the program should wait for samples after each frame. The program can then be sure that all the samples below max size have arrived. The samples that have not arrived by this point will be omitted and not sent. Figure 3.4 illustrates this process. In this figure, there are three different types of objects;

- I denotes the objects that are smaller than max object size and are never omitted.
- II denotes objects that are larger than max object size, but that does not get omitted.
- III denotes objects that are larger than max object size, and that does not arrive before their frame is sent to the FPGA. They are omitted and not sent.

Objects larger than frame size and max object size combined will always be III. If between max object size and frame size + max object size the objects have a chance of getting omitted.

```
421 if (((predictionLine.FrameNumber - mainViewModel.FirstLineNumber) % (Settings.
        Default.ClipFrameSize)) == Settings.Default.MaxObjectSize)
422 {
423     if (Settings.Default.MaxObjectSize < (predictionLine.FrameNumber -
            mainViewModel.FirstLineNumber)) // We need one complete frame first
424     {
425         FrameFinished();
426     }
427 }
```

Listing 3.4: The code for checking if the length of max object size has passed since the previous frame

The code in listing 3.5 shows how the objects are first copied from a list based on having pixel top above the end of the current frame. They are then removed from the list. The copied objects are then sorted according to PixelTop, then PixelBottom then after the one already there if those parameters are both equal.

```
598 lock (classifiedObjectsBlock)
599 {
600     foreach (ClassifiedObject co in classifiedObjects.FindAll((obj) => obj.
            PixelTop < (Settings.Default.ClipFrameSize + mainViewModel.
            CurrentFrameLineNumber)))
601     {
602         classifiedObjectsCopy.Add(new ClassifiedObject(co));
603     }
604
605     classifiedObjects.RemoveAll((obj) => obj.PixelTop < (Settings.Default.
            ClipFrameSize + mainViewModel.CurrentFrameLineNumber));
606 }
607 classifiedObjectsCopy.Sort(comparerClassifiedObject); // Sort first after
        PixelTop, then PixelBottom, then if equal put after.
```

Listing 3.5: How the classifiedObjects are retreived from shared memory and sorted

Frame size and max object size are values counted in number of lines. Frame size must be larger than max object size. The max object size will usually be given as a distance. A line is related to distance through T_t: The period of the trigger signal., and the speed v as show in (3.1).

$$\text{Line length}[m] = T_t \cdot v \tag{3.1}$$

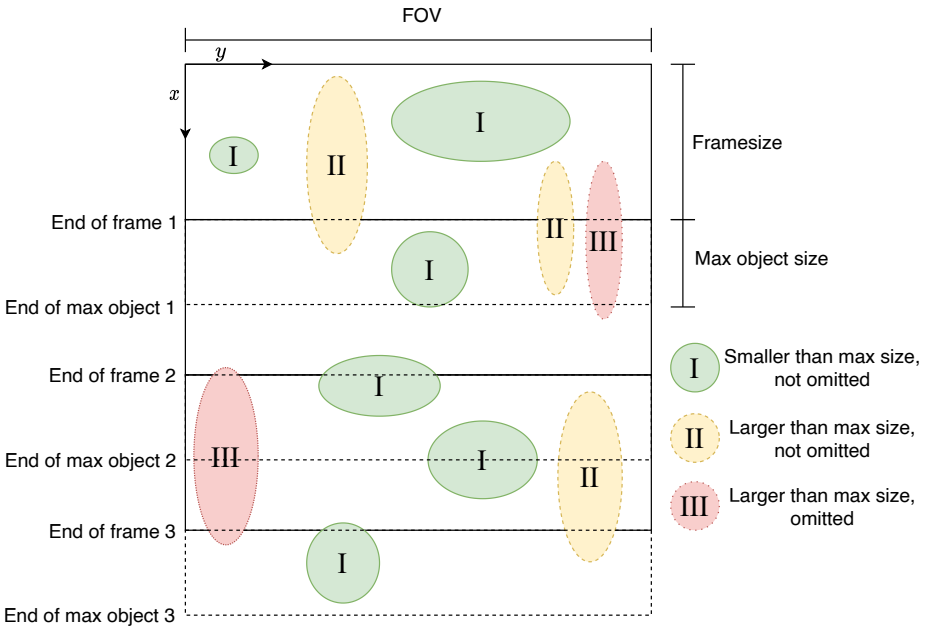


Figure 3.4: A sequence of three frames and how the objects in those frames get sorted.

The number of lines that should be used for a certain max object size is therefore dependent on the period of the trigger signal T_t and on the speed of the conveyor belt v as can be seen in (3.2).

$$\begin{aligned} \text{Max object size}[lines] &= \frac{\text{Max object size}[m]}{\text{Line length}[m]} \\ &= \frac{\text{Max object size}[m]}{T_t \cdot v} \end{aligned} \tag{3.2}$$

If an object is smaller than max object size it is assured to be passed on, if it is larger than frame size and max object size combined it is assured to not being passed on. If it is between those two values there is a probability that it will get rejected as shown by (3.3). The equation is derived from (3.1) and (3.2).

$$p(\text{length}) = \begin{cases} 0, & \text{if } 0 < \text{length} =< maxobjectsize \\ \frac{length - maxobjectsize}{framesize} & \text{if } maxobjectsize < \text{length} < maxobjectsize + framesize \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

3.4.5 Frequency divider verification

Parameters used in this section:

- T_t
- T_e: The period of the event trigger signal.

- LPS

The job of the frequency divider is to send out a trigger signal with a set period T_t to start camera capture, and an event trigger at a set period T_e. T_e is related to T_t by being multiplied by 2^x, where $x \in 0, 1, .., 8$.

To verify how well it performs, an oscilloscope can be connected to the Frequency Divider output. The variance and max deviation in period can then be measured to estimate the stability.

3.4.6 Camera Link verification

A frame grabber is a device that receives recordings image by image from a camera with the camera link protocol. The frame grabber records a timestamp when each image is loaded into memory. This timestamp has an accuracy of 100 ns in the number of digits provided. This timestamp can be tested by running the camera and logging each timestamp. The timestamps can then be plotted against linenumber and analysed.

3.5 Specific sorting system

An FPGA is used to test how the classification module works with third party sorting machines. Communication between SynchSoftware and the FPGA is through a two byte-streams over two TCP/IP connection:

- Port Up, commands from SynchSoftware to the FPGA
- Port Down, responses from the FPGA to SynchSoftware

The event trigger sent to the FPGA is used for time synchronisation. The FPGA creates a timestamp at the moment the pulse from the frequency divider goes from low to high. This timestamp is then sent to SynchSoftware which relates it to the frames containing classified objects received by Breeze Runtime. The timestamps from the FPGA will be referred to as framestamps. SynchSoftware sends commands to start and stop the framestamp stream from the FPGA.

Connections

Figure 3.5 shows how the systems are connected to the computer.

The SWIR-cable contains all the wires used to control the SWIR camera, including the trigger pulse from the Frequency Divider. The event trigger goes out through a coax cable which gets split into two twinned wires. The black wire is ground, and the grey wire carries the event trigger signal. The event trigger then goes into the TTL receiver NI9401[28]. This module receives the event signal with less than 100ns delay and sends it through NI9151[29] to the FPGA. The Camera Link cable is used to transfer the scanned lines from the camera to the frame grabber.

Figure 3.6 shows how these connections look inside the computer.

- The serial card is used to communicate with the Frequency Divider installed below it, and to control a sample mover.
- The Frequency Divider trigger goes into a proprietary cable containing all the control cables for the SWIR-camera. The event trigger goes into a Coax cable and out to NI9401.
- The FPGA is connected to the computer through PCI-Express an trough an NI cable to NI9151. SynchSoftware communicates with the FPGA through PCI-Express while the event trigger is received from NI9151.
- The frame grabber receives scanned lines from the camera and numbers them. Breeze Runtime receives them through PCI-Express.

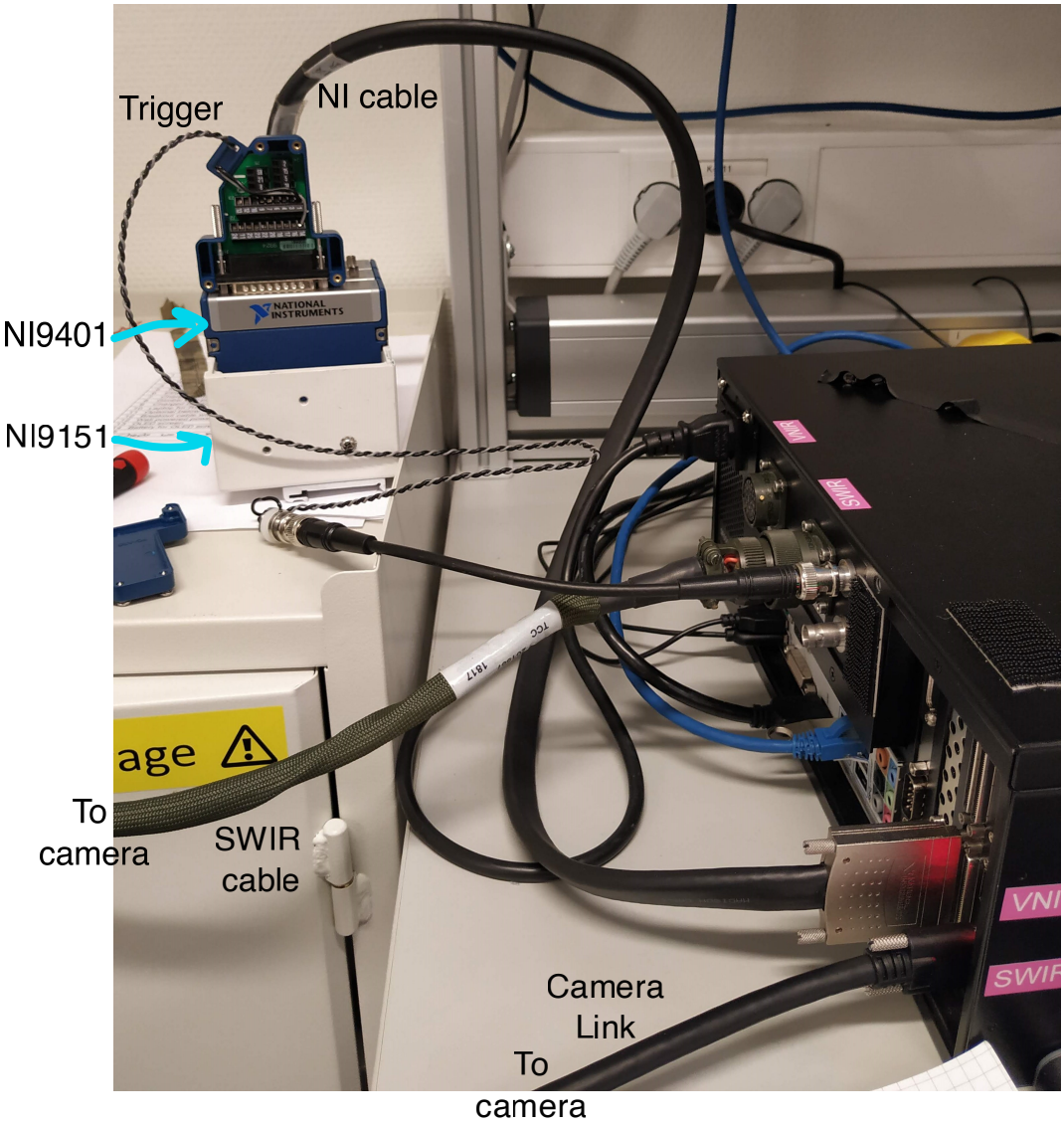


Figure 3.5: A picture of the connections at the back of the computer with labelled cables and components.

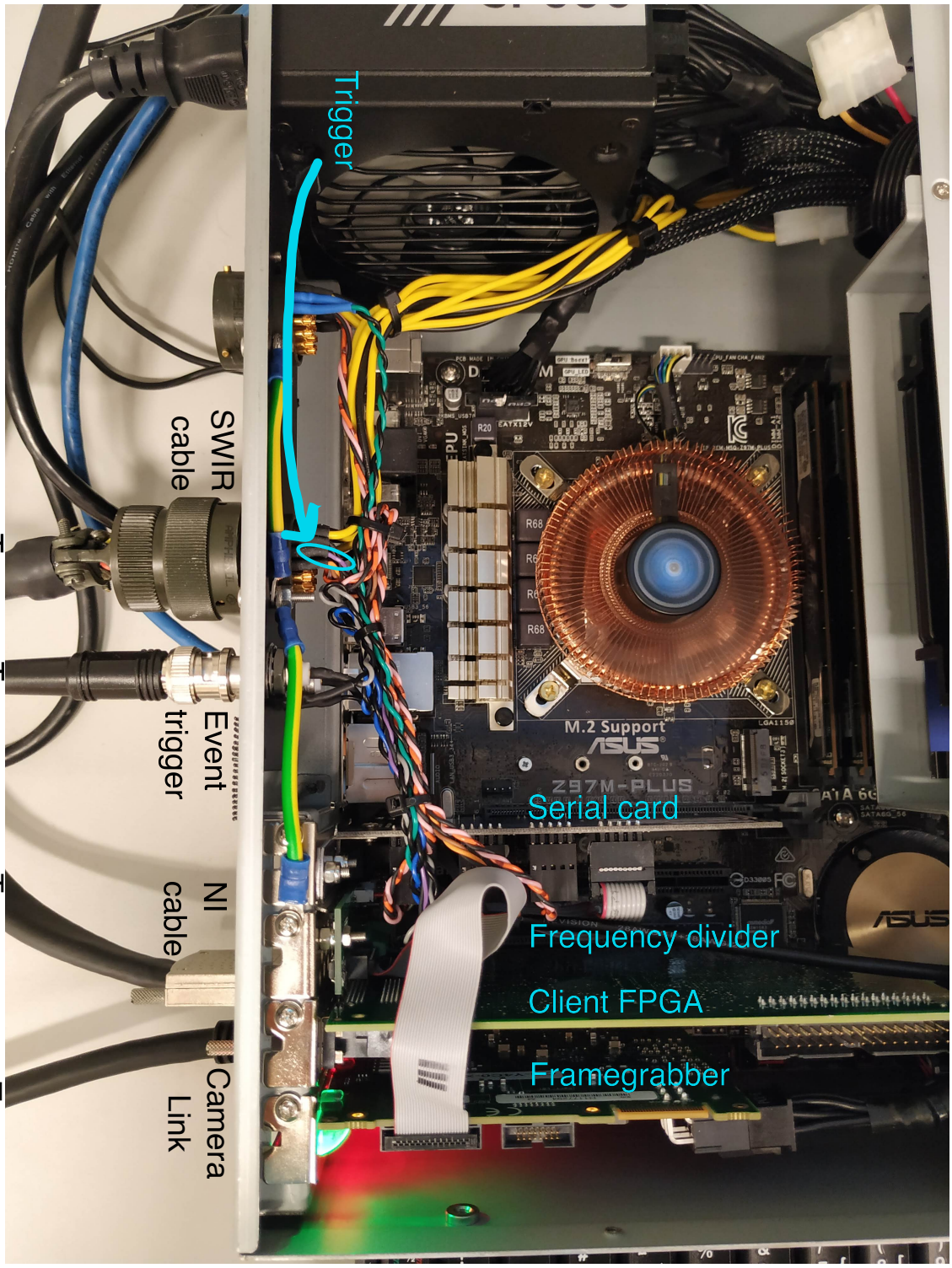


Figure 3.6: A picture of the inside of the computer with labelled connections and components.

Chapter 4

Results

The camera is characterised as part of camera production and the results are summarized in a test report which is included in appendix A.1. Table 4.1 shows a summary of the camera characterisation made by extracting the max values from the graphs. These are absolute max values approximated from the graphs, the graphs should be investigated for accurate estimate of the performance. The FWHM estimate is worst case scenario, as explained in page 10 of the test report: "Note that, when the spectral line hits the centre of a pixel row, then most of the illumination is contained within this row. If the spectral line illumination does not hit the centre of a row, the FWHM plots will appear wider, even though the image quality is very good. This is because the trapezoidal method is used to calculate the FWHM, and if the illumination is evenly distributed between two pixels, the result will be close to 2, even though the physical width of the line is only on the order of 1 pixel."

Property	Value	Unit	Appendix	Comment
Spectral range	[951 - 2515]	nm		
Number of bands	288	#		
Lens WD	912	mm		
Lens Spatial FOV	254	mm		
Spatial pixel resolution (y)	0.661	mm/pixel		
Max spatial FWHM	1.2	pixels	3.1.2	
Max spectral keystone	0.21	pixels	3.2.2	0.04 in the middle
Spectral band resolution	5.4	nm/band		
Max spectral FWHM	1.8	bands	3.3.2	There is one bad pixel
Max spatial smile	0.08	bands	3.4.2	There is one bad pixel
Peak SNR	1151		3.7.2	At 1280 nm
Second Order Suppression	Good		3.6.2	No measurable second order signal
Dynamic range	65536			
Bad pixels	12	#	3.9	These are evenly distributed

Table 4.1: Characterization of the camera based on values from the test report in the appendix as well as information from the settings file of the camera. The max values of spectral FWHM and spatial smile are only measured that high at one wavelength (1442.7nm) and for one pixel (approx. 180), that is why the general value is also included.

Two Breeze Runtime workflows were made. Workflow 1, uses the camera in a mode where it scan lines with all 288 wavelengths. The Breeze Runtime models will then have as much spectral information as possible to analyse.

Workflow 2 uses a windowing function that limits which spectral bands that are recorded. This windowing allows for an increase in framerate that is proportional to the decrease in wavelengths. The processing speed of Breeze Runtime is also increasing with the inverse of the number of wavelengths used. SynchSoftware however, has the same processing speed independent of the number of wavelengths.

The Frequency Divider is then tested to find how stable the trigger signal is. The Camera Link is tested to determine how stable the timestamp it gives is.

SynchSoftware is run through some premature tests at the end of this chapter. Max LPS is found, as well as how SynchSoftware handles objects larger than max object size

Samples

The samples used for the experiments were chosen to be ideal samples for making a proof of concept. Figure 4.1 shows the samples used. The three wood pieces on the left form one group, that PLS-DA should classify differently from the group of three rocks on the right.

Figure 4.1: The six samples recorded, three pieces of wood and three pieces of rock.

The rock samples are homogeneous compounds of sericite, quartz, pyrite and carbonates. The pieces of wood were found outside of NEO and is of an unknown type. They are organic however, which makes their constitutions different from the rock samples. The surface of them was polished down to remove dirt.

4.1 Workflow 1

Equipment used:

- Hyperspectral camera: HySpex SWIR-384[30] with 1 meter lens
- Sample mover: XILab translation stage
- Computer: Windows 10, Intel i7 32Gb Ram
- Recording software: Breeze
- Light source: Industrial halogen lamp
- Reference for normalized reflectance (R_{norm}): White reference Zenith Polymer Diffuse Reflectance sheet, 50%R, SG 3224+SG GLUE-100
- Samples: 3 pieces of wood, and 3 pieces of rock.

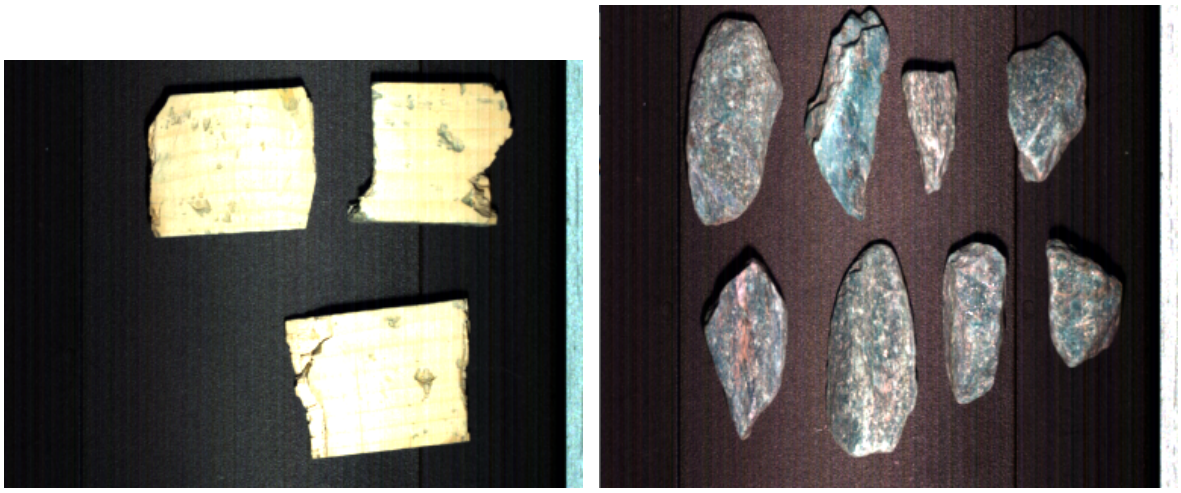
Workflow 1 is going to utilize all wavelengths of the camera.

The following list goes through the steps for making classification models:

1. Choose wavelengths: Changed the settings file to not limit the wavelengths recorded.
2. Connect to camera: Turn the camera on, wait until the camera is sufficiently cooled down, then connect in Breeze.
3. Record the dark and white references: Loaded the file describing the spectrum of the white reference. Place the white reference on the stage and moved the stage while scanning the reference for 200 lines. Closed the shutter and recorded for 200 lines for dark reference.
4. Record training data: The two sets of samples where recorded separately. First eight rocks (only three of the rocks where kept further) and then three pieces of wood. Pseudo RGB images of the recordings are shown in figure 4.2
5. Data clean up and PCA: A separation model was made as described in section 3.3.1.
    - The model creation can be seen in figure 4.3.
    - Figure 4.4 gives a measure of how well the model separates and shows the choice of critical distance.
6. Make a classification model: A classification model was made as described in section 3.3.2.
    - A score plot can be seen in figure 4.5.
    - Figure 4.6 shows the linear regression.
7. Set up workflow: A workflow with the PCA then classification into Wood or Rock, X and Width where set up, it can be seen in figure 4.7.
8. Export workflow to Runtime: The model was exported to the Breeze Runtime folder.

Results from the separation and classification are introduced below.

Figure 4.2 is a pseudo RGB image used to visualize the image. Three wavelengths were chosen to represent the colours red, green and blue. The wood samples look homogeneous except for some grooves in the surface. The rock samples look somewhat inhomogeneous in the image, with different structures across the surface. The wood pieces are bright and look almost saturated, while the rock pieces are quite dark.



(a) Pseudo RGB of wood samples made by representing red with 1218nm, green with 1654nm and blue with 2145nm

(b) Pseudo RGB of Rock samples made by representing red with 1142nm, green with 1905nm and blue with 2014nm

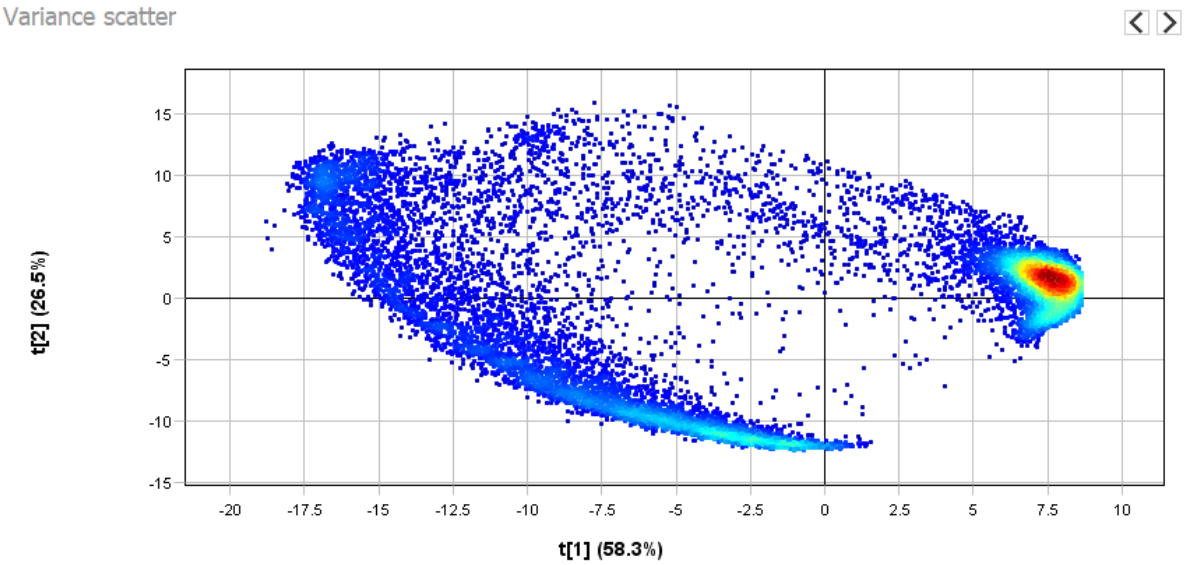
Figure 4.2: The wood pieces are bright and look almost saturated, while the rock pieces are quite dark. The figures are generated by Breeze.

Separation

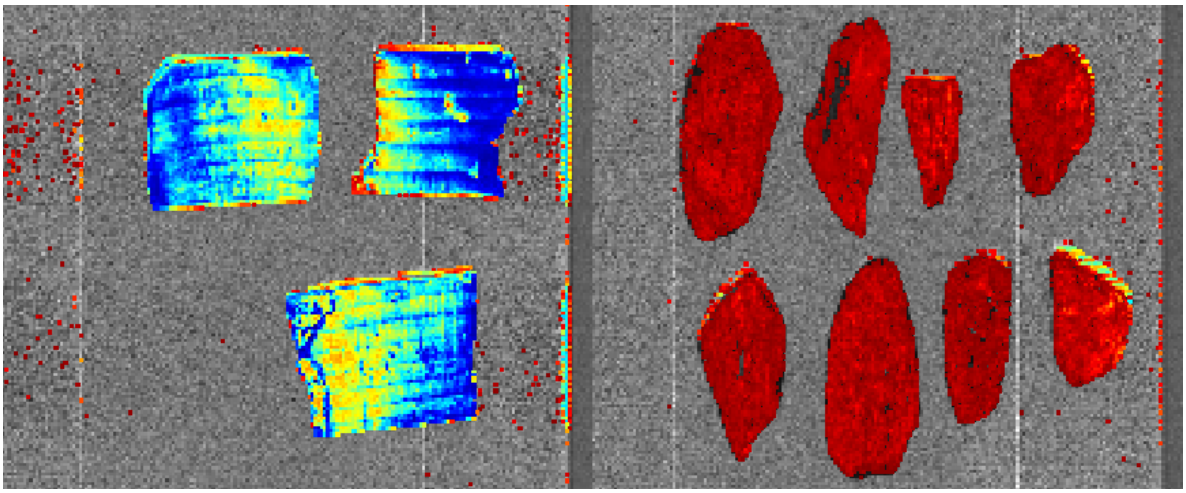
Figure 4.3 shows a variance scatter plot in (a) where the t[1] axis is the first PCA-component, and t[2] describe the second PCA-component. Each dot in the scatter plot represents one of the coloured pixels in (b). (b) consists of two scans put side by side. If the colour of a pixel in (b) is grey-scale, like most of the background, then that pixel is not described by the scatter plot. The colour of non grey-scale pixels relates to the position in the scatter plot. The colour of pixels in (a) is given by their density, blue is the colour for individual pixels while red is the colour for the most dense pixels. There are some background pixels in (b) that are not grey-scale, which can be considered to be noise that is not successfully filtered out.

Figure 4.4 (a) shows a histogram based on the separation from figure 4.3. The grey bars are made out of all the pixels in the "raw" scan, while the black bars are made out of the selected pixels in the plot. They are distributed on the normalised x-axis based on how close to the created PCA model they are. The y-axis shows how many pixels that are included in each bin. The vertical red line is used to choose how different a pixel can be from the model and still be included as a sample. (b) shows how the model with the chosen critical distance works on the original dataset. White colour on a pixel means that a pixel is not included, while a pseudo RGB colour is shown for the pixels that are included. In this figure there are some black pixels spread out where there should only be background. There are also some white pixels inside the pieces of wood that shows that part of them are excluded from the

model. The pieces of rock are almost completely included.

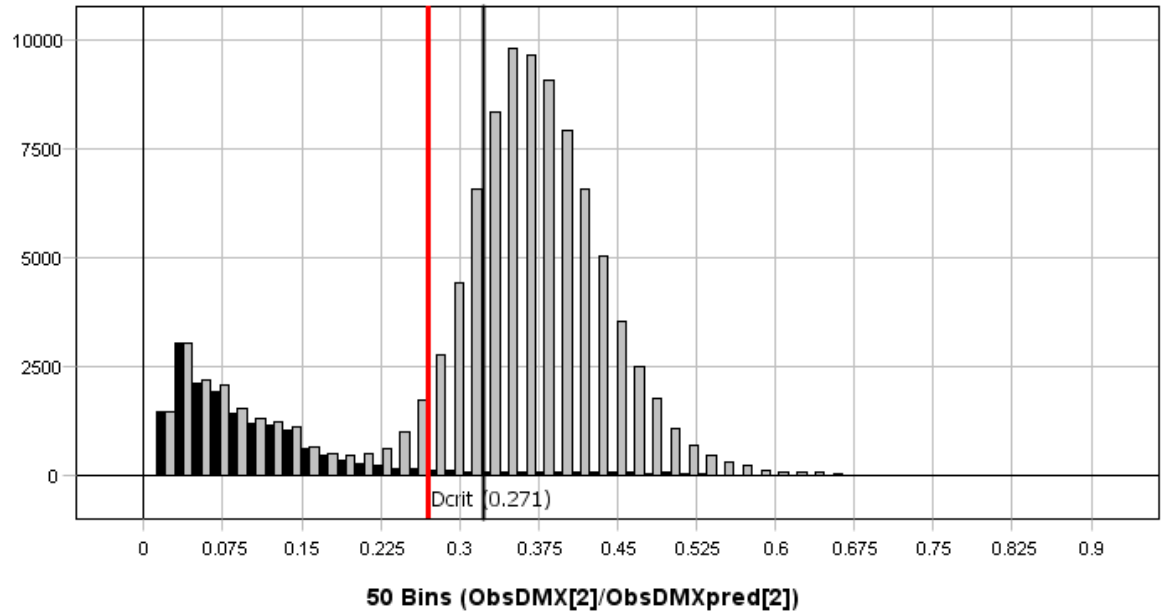


(a) 2D scatter plot. The colours show the pixel density, red is the most dense, while blue is the least dense. The red cluster on the right corresponds to the rocks, while the blue arc on the left corresponds to the pieces of wood.

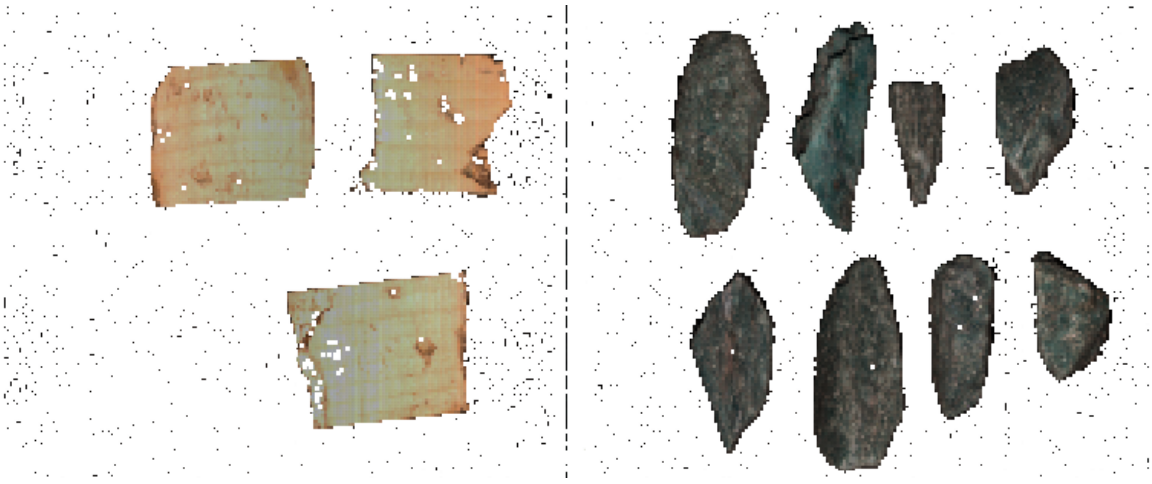


(b) Two scans coloured after where the corresponding pixel in (a) is placed. The left scan is of wood pieces, and the right scan is of rocks.

Figure 4.3: As much as possible of the scatter plot that corresponds to the background has been removed, so that the PCA model should correspond to the two sets of samples. The figures are generated by Breeze.



(a) The grey bars are made out of all the pixels in the "raw" scan, while the black bars are made out of the pixels included in figure 4.3. The x-axis shows a normalised distance to the model. The y-axis shows the amount of pixels in a bin. The red line is the chosen critical distance. The figure is generated by Breeze.



(b) Pseudo RGB with wood pieces on the left, rocks on the right. The white pixels are separated out as background, while the pixels with colour har described by the model.

Figure 4.4: This plot shows how the critical distance is decided. The red line in (a) is moved back and forth and the number of included pixels in (b) changes to aid the decision making. The figure is generated by Breeze.

Classification

Figure 4.5 shows how the rocks and wood pieces are placed in a 2D space made out of the dimensionless first and second PLS-DA-component. The first PLS-DA-component (t[1]) describes 91.4% of the covariance between the samples. The minimum distance between the rock and wood samples along this dimension is 15. The rocks are collected within 2-3 values, and the wood pieces within 5 values. The second PLS-DA-component describes 7.98% of the covariance. The rocks are collected within 4 values, while the wood pieces are spread out within 12 values.

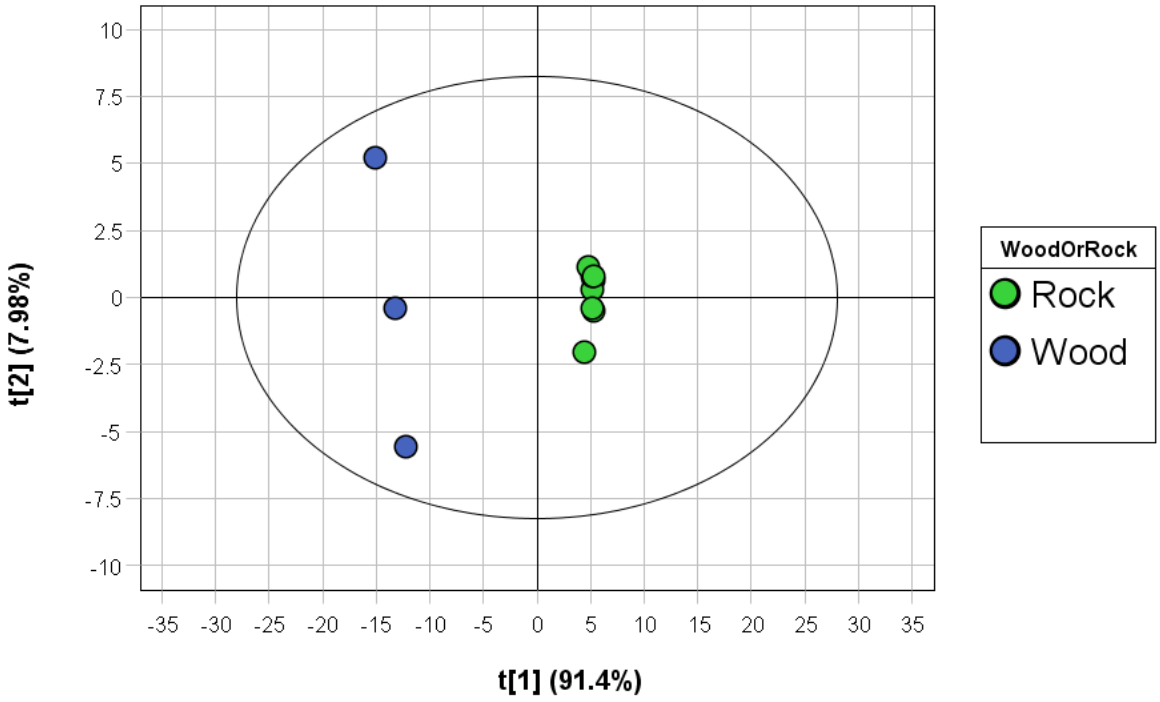


Figure 4.5: Score plot with dimensionless axes for the classification. The green pieces of rocks are collected in both directions while the pieces of wood are collected in the t[1] direction and spread out in the t[2] direction. The linear separation model probably only needs t[1]. The figure is generated by Breeze.

Figure 4.6 shows the linear model made with the first two components of the PLS-DA-model. The linear regression have quality parameters: $R^2 = 0.9976$ and $RMSE = 0.0216$. R^2 is a value between -1 and 1 that describes how close the values are to the model. $RMSE$ is a measure for the error of the estimation.

Workflow

The combined workflow can be seen in figure 4.7. The Measurements are first sent through a Sample model called Rock vs Wood. This is the PCA-model that separates the pixels corresponding to rocks and wood samples from the pixels corresponding to background. It then finds the clusters of these that are larger than 600 pixels. The value corresponding to the leftmost part of the objects (the X value) and the Width of the objects are extracted. The objects are sent to the PLS-DA-model called WoodOrRock which separates them into the two groups Wood and Rock.

Running the separation in real-time

Figure 4.8 shows the real time separation of the test samples on the sample mover. This is a screen shot of a live scan where the pixels are classified 2-3ms after the camera scans them. The green rocks are all almost fully included in the model with no missing pixels. The blue pieces of wood have missing pixels, especially the leftmost piece. All the wood and the rock pieces are found, as they are marked by a white border with an X at the middle of the samples. The leftmost rock is not identified as an object yet, due to not being scanned in its entirety.

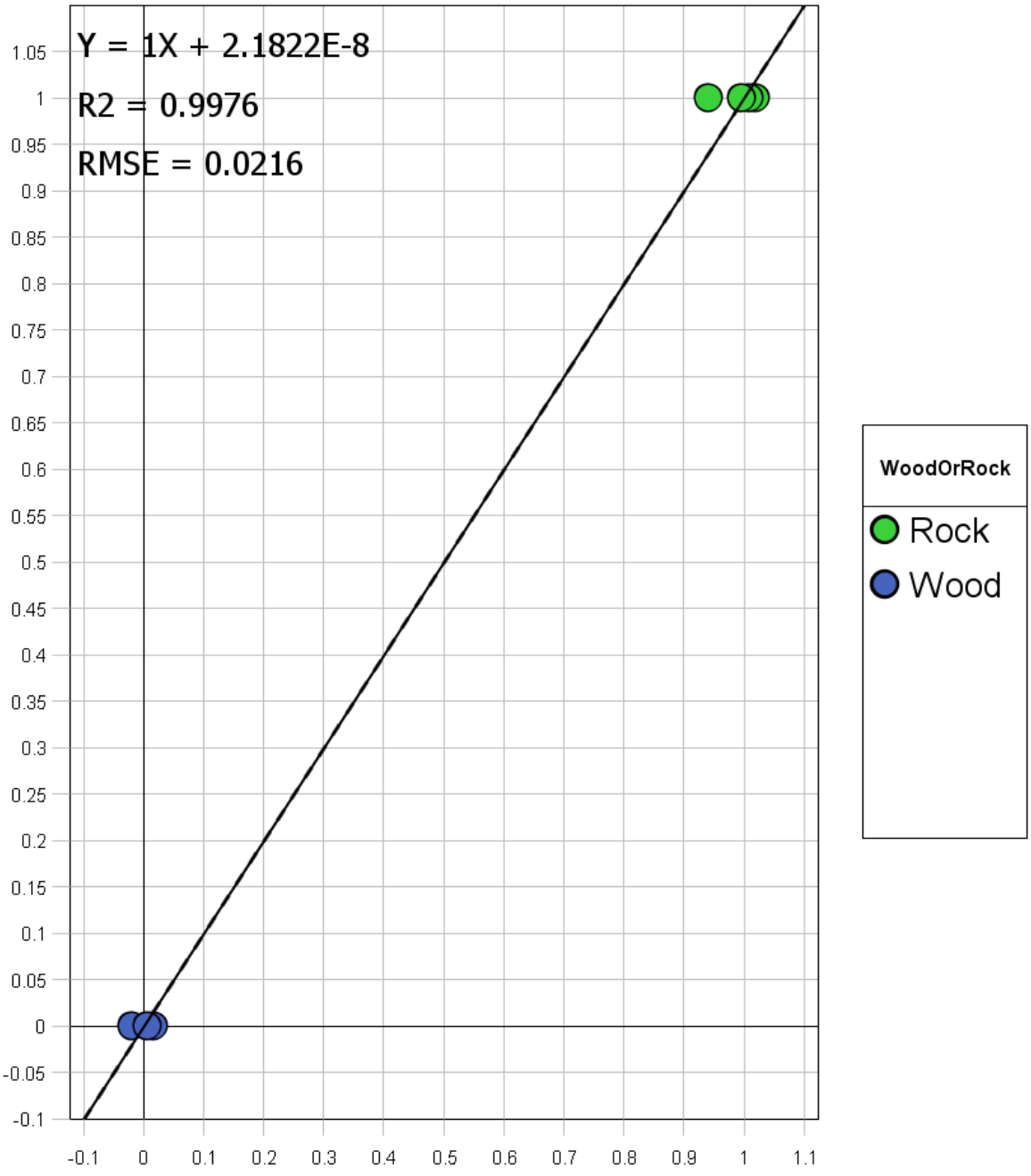


Figure 4.6: The linear regression line formed on the dimensions provided by the PLS-DA-model. The figure is generated by Breeze.

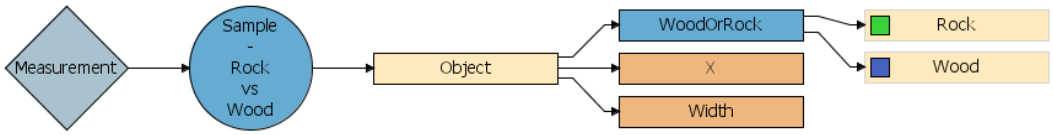


Figure 4.7: Breeze Runtime receives the measurements from the camera, finds the samples with PCA and classifies them with PLS-DA. The output from Breeze Runtime for each object is: X, Width and the classification of rock or wood. The schematic is generated by Breeze.

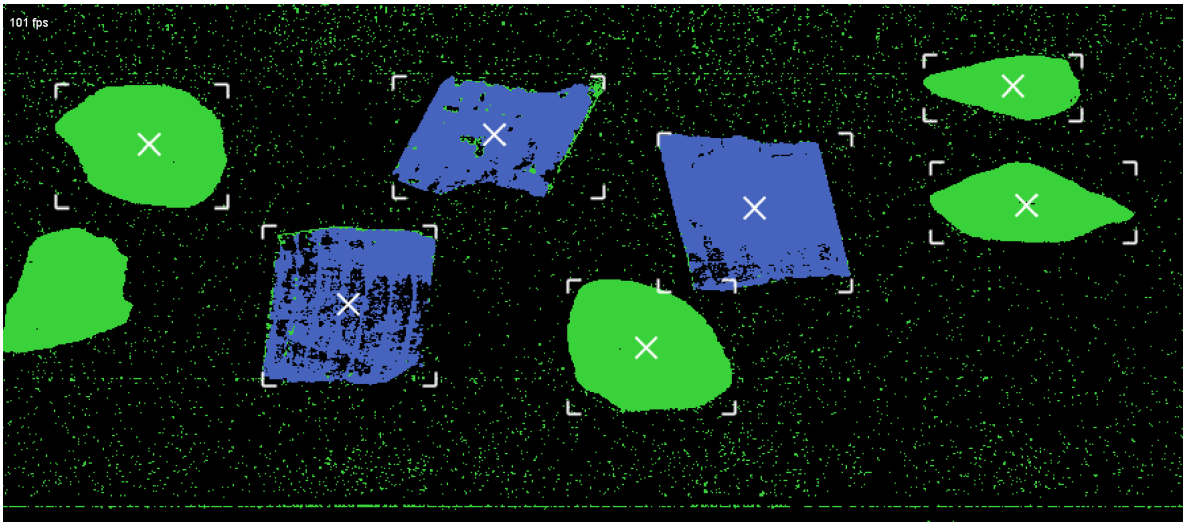


Figure 4.8: Testing how the real time separation of the training samples work. Blue pixels are classified as wood, green as rocks. Taken at 101 LPS. The green rocks are all almost fully included in the model with almost no missing pixels. The blue pieces of wood have missing pixels, especially the leftmost piece. All the wood and the rock pieces are found, as they are marked by a white border with an X at the middle of the samples. The background has many individual pixels that are classified as rocks, but not in large enough clusters to be thought to be samples. Screenshot of live action from Breeze.

4.2 Workflow 2

This workflow uses the same equipment as in workflow 1, but the wavelengths recorded by the camera will be limited.

1. Choose wavelengths: Documented right after this list under "limiting the number of wavelength".
2. Connect to camera: Turned the camera on, waited until sufficiently cooled then connected in Breeze.
3. Record the dark and white references dark and white references: Loaded the file describing the spectrum of the white reference. Placed the white reference on the stage and moved the stage while scanning the reference for 200 lines. Closed the shutter and recorded for 200 lines for dark reference.
4. Record training data: The two sets of samples were recorded separately. First three rocks and then three pieces of wood.
5. Data clean up and PCA: A separation model was made as described in section 3.3.1.
   - The model creation can be seen in figure 4.12.
   - Figure 4.4 gives a measure of how well the model separates and shows the choice of critical distance.
6. Make a classification model: A classification model was made as described in section 3.3.2.
   - A score plot can be seen in figure 4.14.
   - A showing the linear regression can be seen in figure 4.15.
7. Set up workflow: A workflow with the PCA then classification into Wood or Rock, X and Width where set up, it can be seen in figure 4.16.
8. Export workflow to Runtime: The model was exported to the Breeze Runtime folder.

Limiting the number of wavelengths

To increase the max LPS of the camera, and the processing speed of Breeze Runtime, the camera will be limited to a certain number of wavelengths. A feasible goal is to run the camera at $LPS = 2000$Hz. Max LPS is increased proportionally with the decrease in wavelengths utilized. When using all wavelengths the max speed is 399Hz. To achieve 2000Hz, a little under a fifth of the wavelengths should be used. (4.1) shows the full number of wavelengths used multiplied with the ratio of current and wanted number of wavelengths. The product is the max number of wavelengths that can be used at 2000Hz. 56 wavelengths will be used instead to have som extra margin.

$$288\text{wavelengths} \cdot \frac{399Hz}{2000Hz} \approx 57.4 \tag{4.1}$$

When decreasing the number of wavelengths the whole workflow is redone. New recordings and models are made using the new wavelengths. This is due to a practical reason; it is harder to coordinate Breeze and the camera to know which wavelengths in a training scan that correspond to which wavelength in a test scan, than making new scans with the correct number of wavelengths.

These wavelengths were chosen based on experiences made during workflow 1. The most important wavelengths for the decision making in a PLS-DA model can be identified when making it. Figure 4.9 shows the most important wavelengths for the classification.

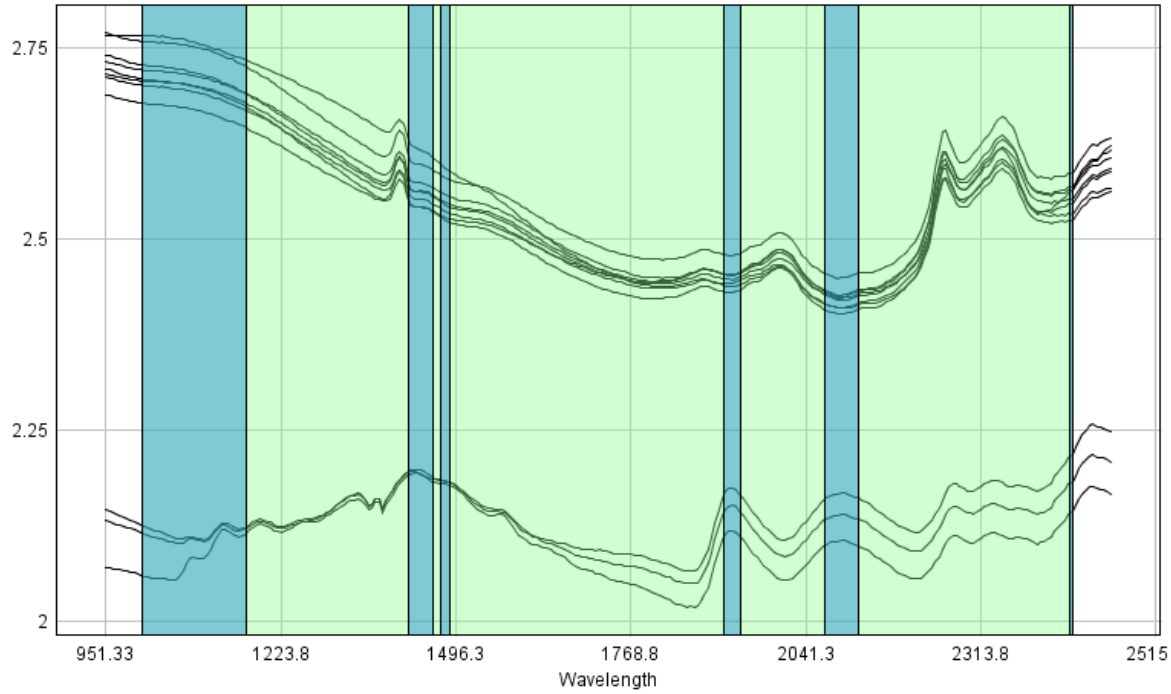


Figure 4.9: A plot of the spectra from the samples using all the wavelengths. The x-axis is λ, the y-axis is dimensionless. The bottom three lines are from the wood pieces, and the top eight are from the rocks. The vertical blue bars marks the 56 most important wavelengths for the classification. Figure generated with Breeze.

The camera was configured with these wavelengths, but they turned out to not give a good separation of samples and background. This was assumed to be due to the workflow 1 PLS-DA not containing information about the background. The most important wavelengths for that model were therefore not necessarily good ones for separation between the samples and background. To find wavelengths that could separate from the background, the background spectra was analysed. Figure 4.10 shows the spectra of part of the background compared to the spectra from part of a wood sample. The background has a monotonous downward slanting slope. This spectra was approximated to be straight.

With the knowledge that the background is straight and downward slanting, the wavelengths could be hand picked to those showing features in the wood and rock spectra in figure 4.9.

Figure 4.11 shows the 56 wavelengths chosen marked with blue. More of the noticeable features are chosen this time. The rightmost wavelengths are the biggest difference. The rock spectra makes an upward turn around 2200nm, which is the opposite of the backgrounds downward slant.

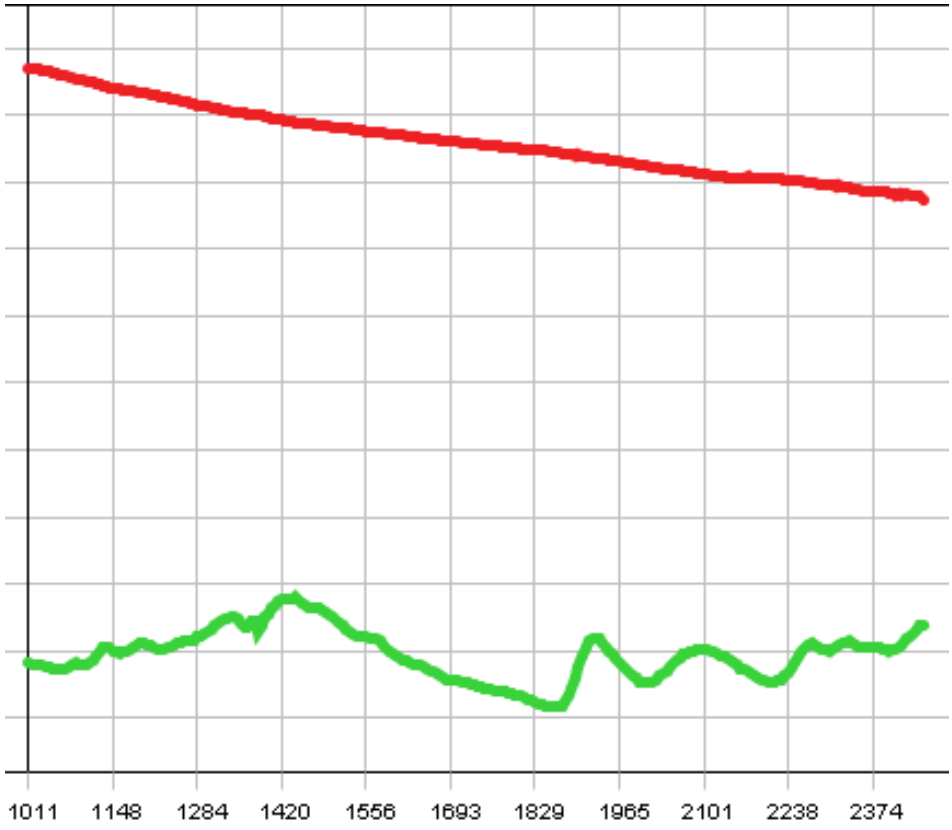


Figure 4.10: Normalised reflectance spectra, red line is a selection from the background, and the green line is a selection from a wood sample. The x-axis is wavelength, and the y-axis is normalised reflectance. The figure is generated with Breeze.

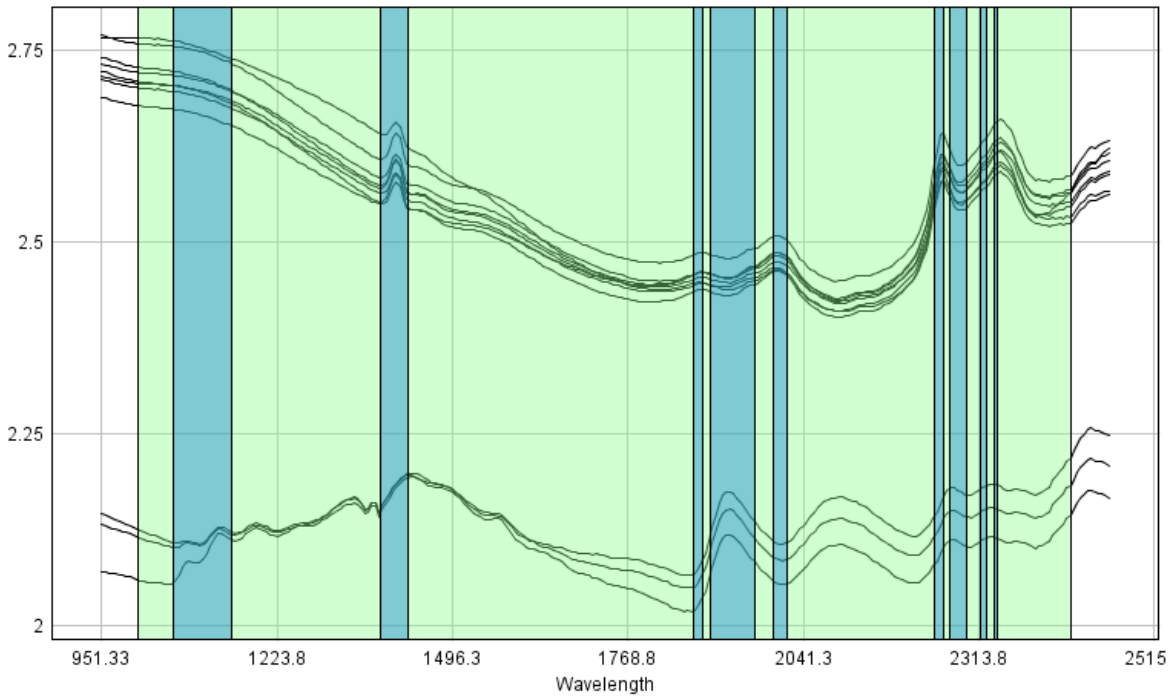
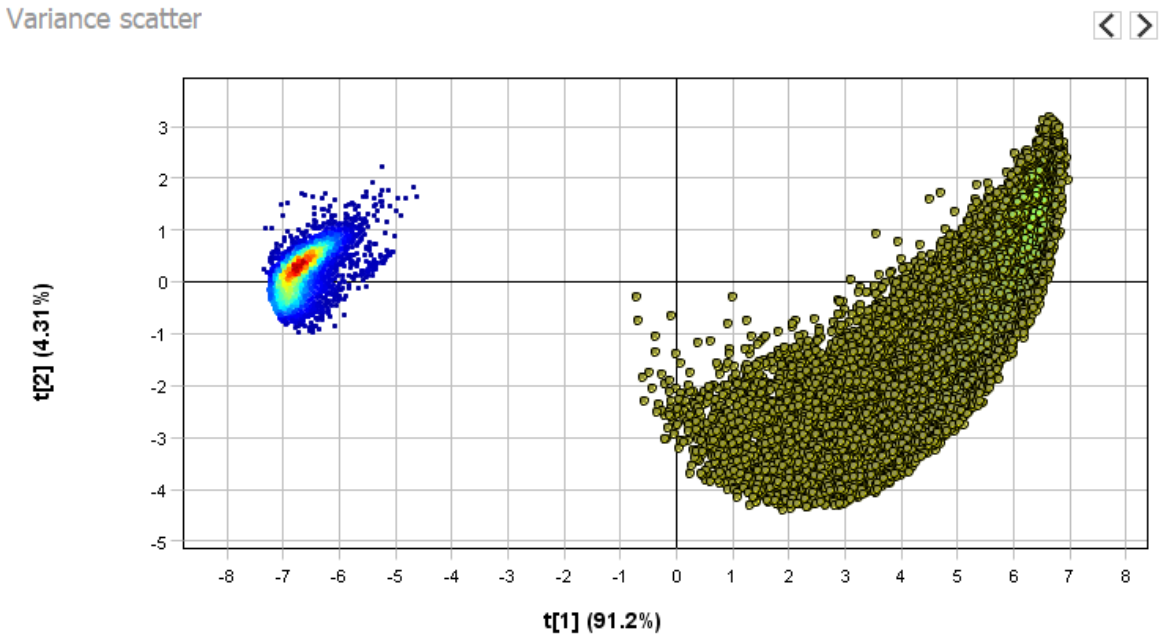


Figure 4.11: The 56 wavelengths chosen for making this workflow. They are marked by the blue columns out of the 288 wavelengths. The x-axis is λ, the y-axis is dimensionless. The bottom three lines are the wood pieces, and the top eight are the rocks. The figure is generated with Breeze.

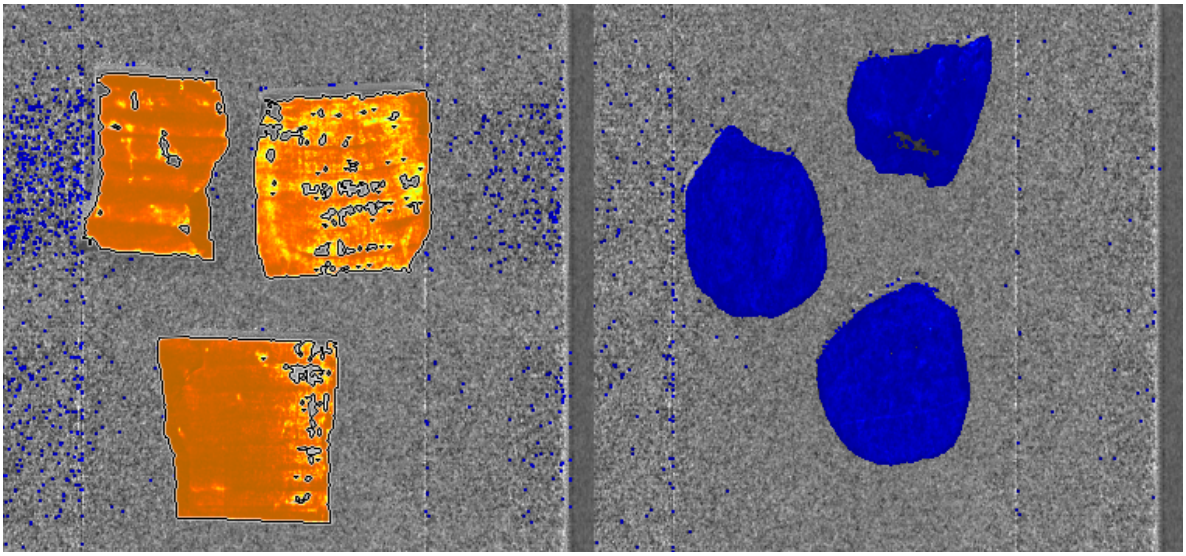
Separation

Figure 4.12 shows a variance scatter plot in (a) where the t[1] axis is the first PCA-component, and t[2] describe the second PCA-component. Each dot in the scatter plot represents one of the coloured pixels in (b). (b) consists of two scans put side by side. If the colour of a pixel in (b) is grey-scale, like most of the background, then that pixel is not described by the scatter plot. The colour of non grey-scale pixels relates to the position in the scatter plot. The colour of pixels in (a) is given by their density, blue is the colour for individual pixels while red is the colour for the most dense pixels. The graph has a tight cluster on the left which corresponds to the rocks, and the selected cluster on the right corresponds to the pieces of wood. There are some background pixels in (b) that are not grey-scale, especially around the pieces of wood. These can be considered to be noise that is not successfully filtered out.

Figure 4.13 (a) shows a histogram based on the separation from figure 4.12. The grey bars are made out of all the pixels in the "raw" scan, while the black bars are made out of the selected pixels in the plot. They are distributed on the x-axis based on how close to the created PCA model they are. The vertical red line is used to choose how different a pixel can be from the model and still be included as a sample. (b) shows how the model with the chosen critical distance works on the original dataset. White colour means that the pixel is not included, while a pseudo RGB colour is shown for the pixels that are included.

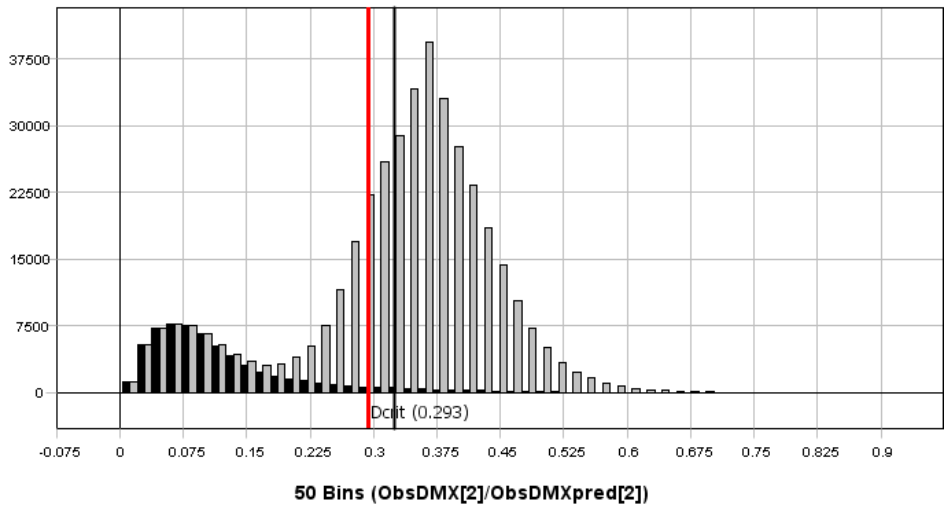


(a) 2D scatter plot. The cluster on the right is selected manually.

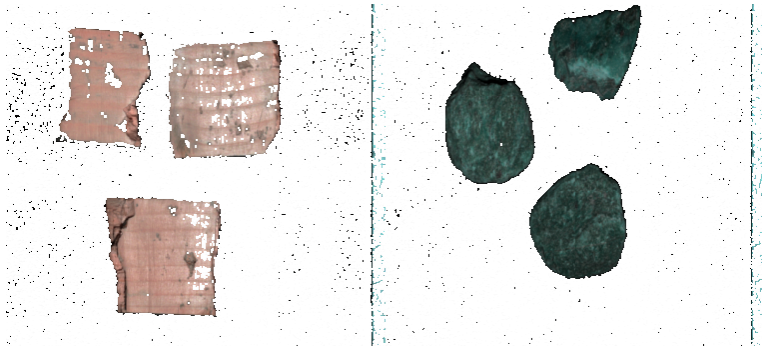


(b) Wood pieces on the left, rocks on the right. Notice that most of the wood pieces are outlined due to the selection made in the scatter plot.

Figure 4.12: As much as possible of the scatter plot that corresponds to the background has been removed, so that the PCA model should correspond to the two sets of samples. The parts of the scatter plot that corresponds to the wood pieces are selected. Figures generated with Breeze.



(a) Critical distance



(b) Pseudo RGB with wood pieces on the left, rocks on the right.

Figure 4.13: The red line marks the chosen critical distance and the corresponding separation in the scan. The white pixels are separated out as background. The figures are generated with Breeze.

Classification

Figure 4.14 shows how the rocks and wood pieces are placed in a 2D space made out of the dimensionless first and second PLS-DA-component. The first PLS-DA-component (t[1]) describes 97.2% of the covariance between the samples. The minimum distance between the rock and wood samples along this dimension is 10. The rocks are collected within 1, and the wood pieces within 3 values. The second PLS-DA-component describes 2.22% of the covariance. The rocks are collected within 0.5 values, while the wood pieces are spread out within 4 values.

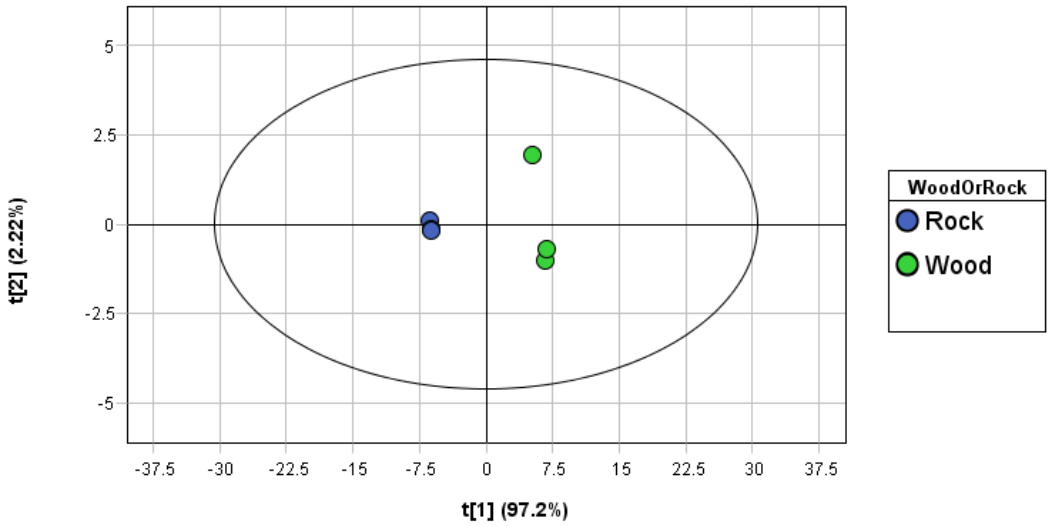


Figure 4.14: Score plot with dimensionless axes for the classification. The blue pieces of rocks are collected within 1 value in both directions. The wood is collected in the t[1]-direction, but spread out in the t[2] direction. The figure is generated with Breeze.

Figure 4.15 shows the linear model made on the first two components of the PLS-DA-model and the corresponding R^2 and RMSE values.

Workflow

The combined workflow can be seen in figure4.16. The Measurements are first sent through a Sample model called RockAndWood56Wavelengths. This is the PCA-model that separates the pixels corresponding to rocks and wood samples from the pixels corresponding to background. It then finds the clusters of these that are larger than 600 pixels. The value corresponding to the leftmost part of the objects (the X value) and the Width of the objects are extracted. The objects are sent to the PLS-DA-model called WoodOrRock which separates them into the two groups Wood and Rock.

Running the separation in real-time

Figure 4.17 shows the real time separation of the testing objects on the sample mover. This is a screen shot of a live scan where the pixels are classified 2-3ms after the camera scans them. The blue rocks are all fully included in the model with only a few missing pixels. The green pieces of wood have

missing pixels, especially the two pieces to the right. All the wood and the rock pieces are found, as they are marked by a white border with an X at the middle of the samples. The two rocks to the left ate not identified as objects yet, due to not being scanned in their entirety. Due to a large collection of misclassified pixels at the bottom, three clusters there are bigger than 600 pixels and gets identified as objects. They are outside the edge of the sample mover.

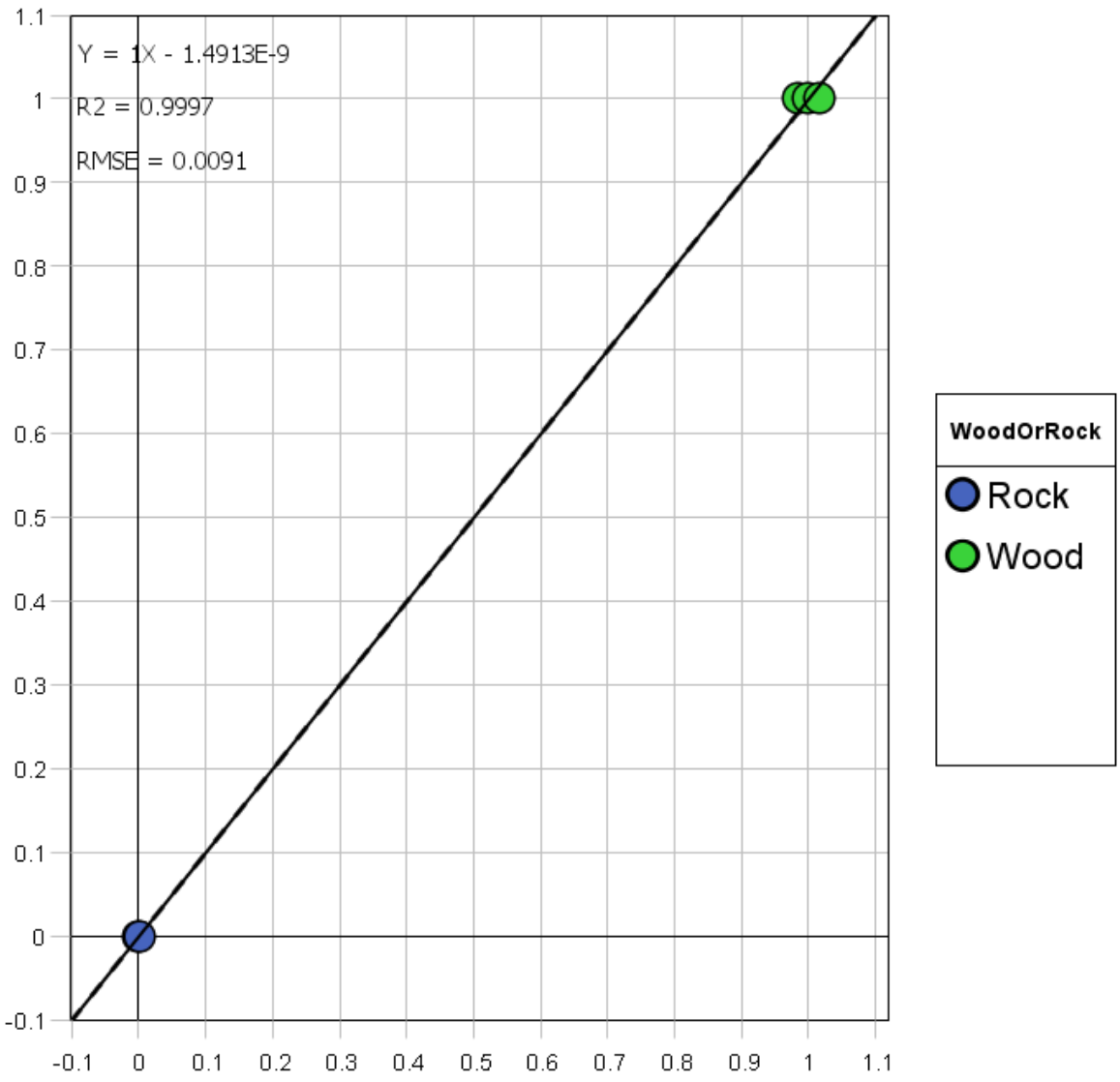


Figure 4.15: The linear regression line formed on the dimensions provided by the PLS-DA-model. The figure is generated with Breeze.

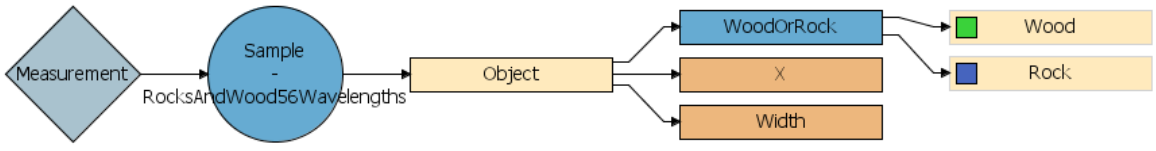


Figure 4.16: Breeze Runtime receives the measurements from the camera, finds the samples with PCA and classifies them with PLS-DA. The output from Breeze Runtime for each object is: X, Width and the classification of rock or wood. The schematic is generated by Breeze.

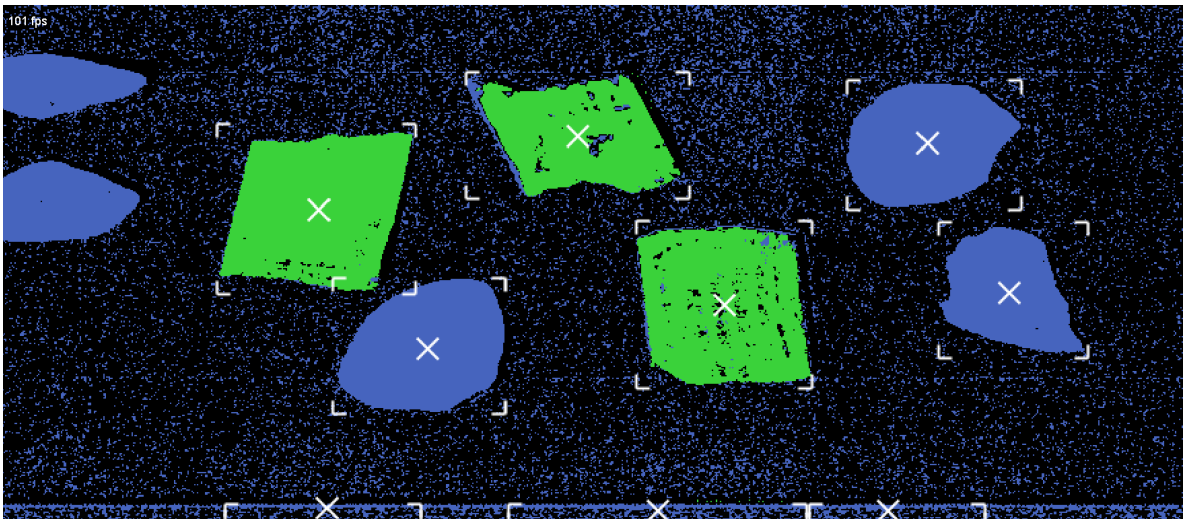


Figure 4.17: Testing how the real time separation of the training samples work. Green pixels are classified as wood, blue pixels as rock. Taken at 101 LPS. The blue rocks are fully included in the model with just a few missing pixels. The green pieces of wood have missing pixels, especially the two on the right. All the wood and rock pieces are found, but three extra objects are found on the edge of the sample mover. Screenshot of live action from Breeze

4.3 General sorting system

4.3.1 frequency divider

To verify the stability of the frequency divider an oscilloscope was used to measure the variance and max min frequency value of the trigger and event trigger.

The oscilloscope used is the USB oscilloscope Picoscope 2000A[31] with 25 MHz bandwidth. Table 4.2 shows the result of these measurements. The sample interval depended upon the measurement window, and is therefore longer when measuring signals with longer periods.

	Period[ms]					Sample
Output	min	avg	max	σ[ns]	#Cycles	interval[ns]
Trigger	5.064	5.064	5.064	287.2	961	640
Event	648.2	648.2	648.2	37590	961	81920
Trigger	0.9998	1.000	1.000	55.10	961	160
Event	128.0	128.0	128.0	10360	961	20480

Table 4.2: Values read from the trigger and event trigger outputs of the frequency divider with a Picoscope 2000A.

4.3.2 Camera link

The stability of the frame grabber timestamp was investigated by running a SWIR-384 camera in a free-run mode (using an internal oscillator for triggering) and recording the timestamps for each image. Figure 4.18 shows two plots, the top one showing the difference in time between two consecutive timestamps and the bottom one just showing the timestamps of consecutive lines. Both the plots have been focused on problem areas and shows the biggest non-linearity that where found during the test period. The timestamp have the unit of 100 ns.

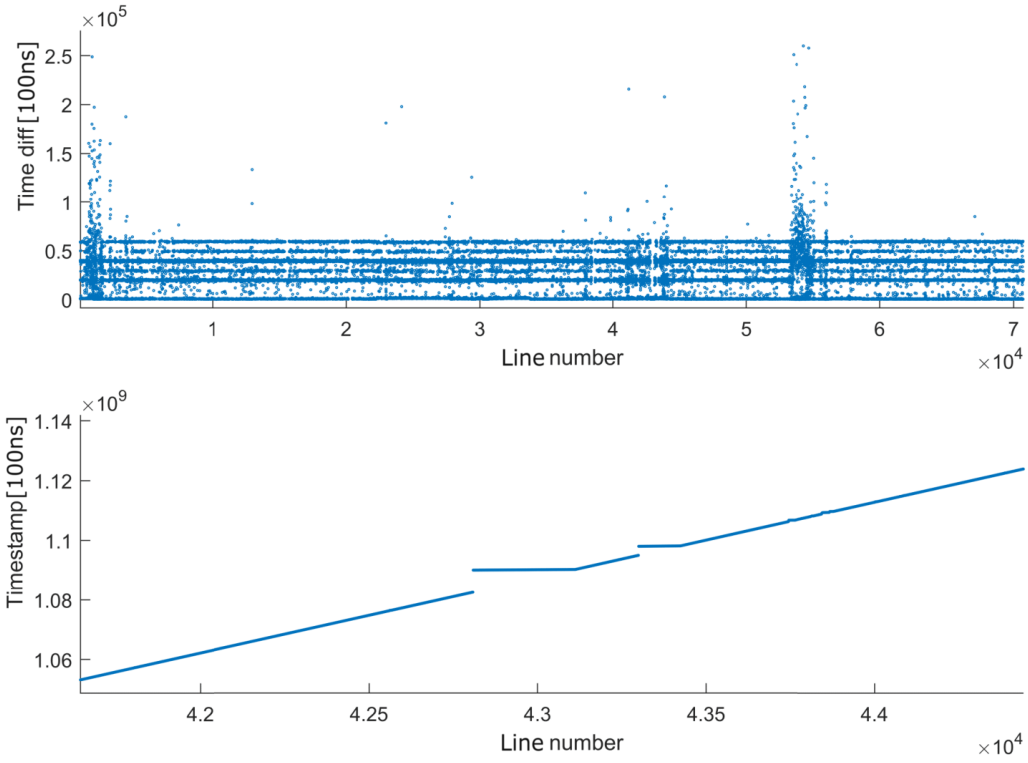


Figure 4.18: The results after a stability test. The camera was run in a free-run mode at 395Hz while the timestamps where logged. The top graph shows the difference between consecutive timestamps, they should have all been at approximately $0.253 \cdot 10^5$. The bottom graph shows the timestamps sequentially, it should have been a straight line, but there are several gaps. Both graphs were focused in on problem areas.

4.4 Specific sorting system

The components used in the lab setup are listed below:

- Frame grabber (Camera Link card): Xtium-CL_MX4_1
- Camera Link cable
- Stage: Standa translation stage, lab.
- Frequency divider v8.1
- Components for sending trigger pulse from Frequency Divider to client FPGA
  - Coax cable
  - Twisted pair cable with female coax in one end.
  - NI9924 Front-mount terminal block for 25-pin D-Sub Modules, SN: HC8033087
  - NI9401 8-Channel, 100 ns, TTL Digital Input/Output Module, SN: HC8068465
  - NI9151 4-Slot, Reconfigurable I/O Module CompactRIO Chassis

  - NI9151 cable
- Client FPGA
- HySpex SWIR-384, SN3135

With the following settings:

- $v = 4\text{cms}^{-1}$
- $LPS = 50\text{Hz}$
- frame size = 128

SynchSoftware logs everything it does as JSON-lines which can be explored afterwards to see how, and how well the system performed. The following information is a product of such exploration done in Jupyter Notebook. Contact the author for the python code from this exploration.

Max object size

Some objects were not passed on to the FPGA even though they were of the type to reject. SynchSoftware is programmed to wait for max object size number of lines after receiving the last line from a frame before concluding that the frame is complete. If the objects are larger than this number of lines, then they have a chance of getting omitted, i.e. not sent to Client. Figure 4.19 shows how this can look when having a max object size of 50, while the mean object size is approximately 60. The objects that are not passed on are larger than max object size and arrives so late in its frame, that the bottom line is later than max object size.

Max LPS

When running the program at more than LPS 125Hz an exception started getting thrown by SynchSoftware. The exception was a KeyNotFoundException that said that the linenumbers for the first and last line in objects was not available in the predictionLineDictionary. The timestamp for when a line is received by the frame grabber is compared the timestamp for when a line is loaded by SynchSoftware in figures 4.20 - 4.22. In figure 4.20 the line for ReceivedBySynchSoftwareTime is hidden behind the one for CaptureTime. This shows that there is little delay between the acquisition of a line and the processing in SynchSoftware at LPS = 100Hz. At LPS = 125Hz in figure 4.21 the lines are mostly overlapping except for between linenumber 18400 and 20000. In between those lines the delay of ReceivedBySynchSoftwareTime first increases in two bursts, and then the delay is slowly being processed away. Figure 4.22 shows the situation at LPS 227Hz. SynchSoftware does not finish processing a predictionLine before the next one arrives. The delay between them is continuously growing.

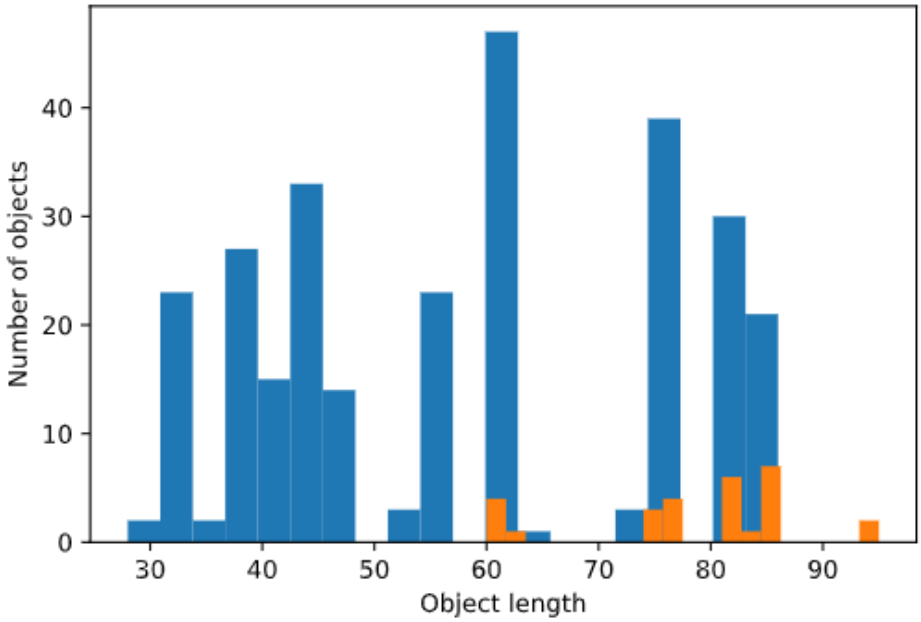


Figure 4.19: Histogram rejected objects

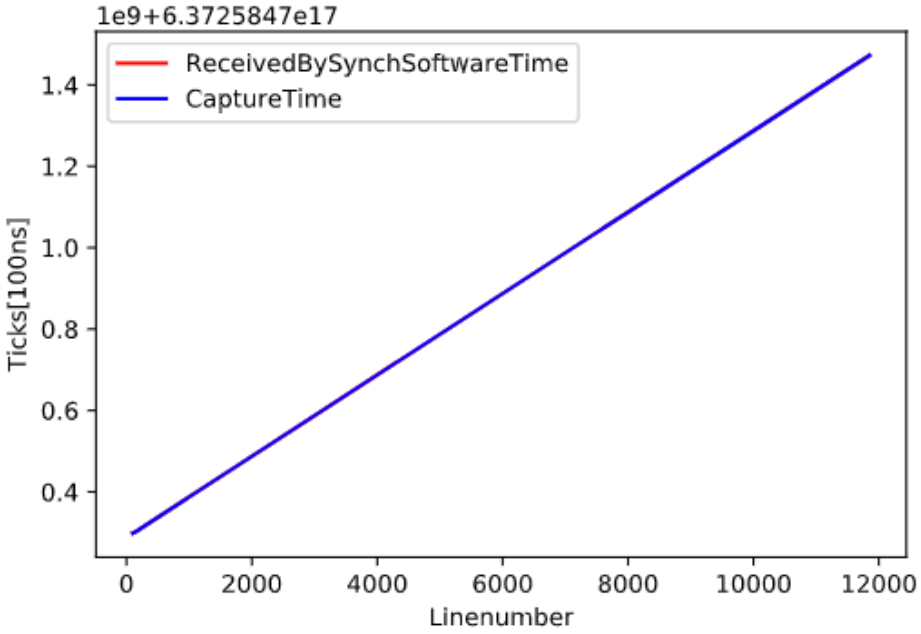


Figure 4.20: Comparison of the time when the frame grabber loads the line, and when the line is received by SynchSoftware at a period of 10000μs (LPS = 100 Hz). The line for ReceivedBySynch-SoftwareTime is completely hidden behind the one for CaptureTime showing little delay.

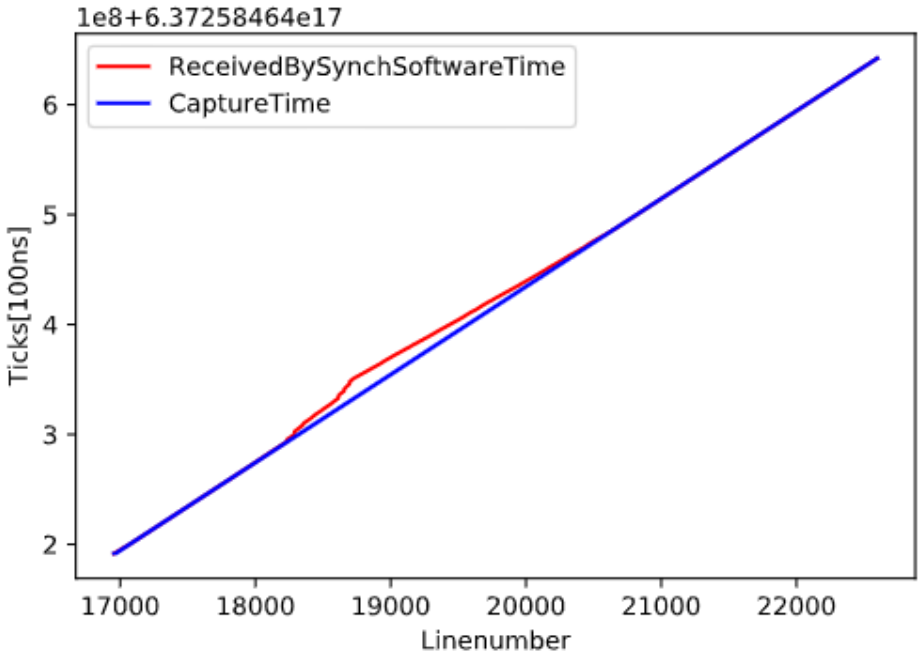


Figure 4.21: Comparison of the time when the frame grabber loads the line, and when the line is received by SynchSoftware at a period of 8000μs (LPS = 125 Hz). The line for ReceivedBySynchSoftwareTime shows delay building up in two steps, and then the lines slowly closing in on each other. The computer can barely keep up with the processing of predictionLines.

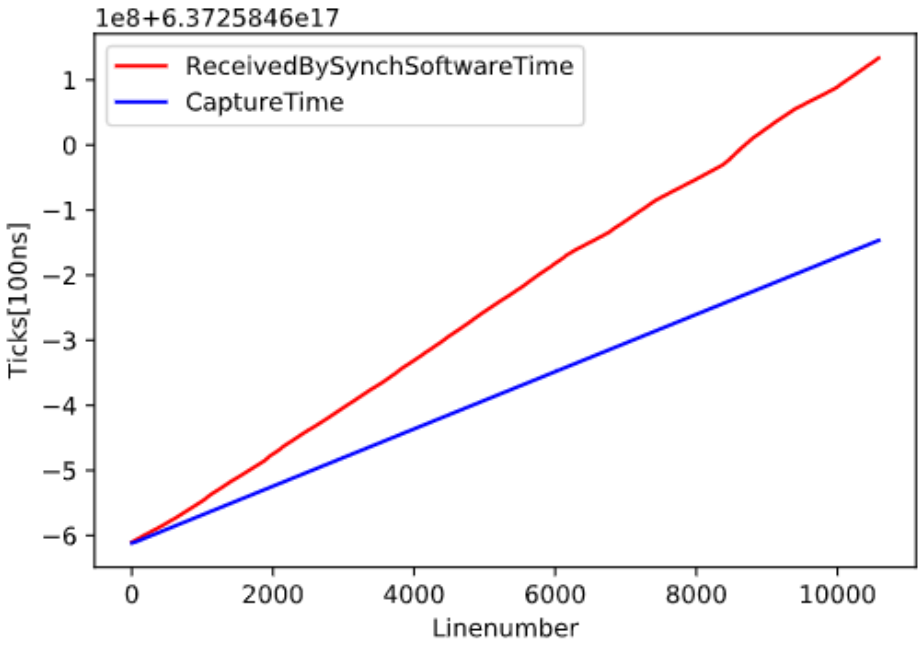


Figure 4.22: Comparison of the time when the frame grabber loads the line, and when the line is received by SynchSoftware at a period of 4400μs (LPS 227 Hz). The line for ReceivedBySynchSoftwareTime shows delay building up continuously. The computer cannot finish the processing of one predictionLine before the next one arrives.

Chapter 5

Discussion

5.1 Test report review

Table 4.1 lists parameters that describe how the camera performs. Lens Spatial FOV says that the camera can only record a width of 254mm. This FOV is narrow compared to normal conveyor belts with widths of one to two meters. At the same time, the spatial pixel resolution of 0.661mm/pixel is excellent. A sorting system that uses 110 nozzles per meter, or 9.1mm per nozzle, was used as a benchmark. Compared to that system, the camera has 13 times as high resolution. A view expander could be used to utilize the available pixels better. That is a lens with a wider angle than the current 16° and would increase the FOV. For example, a 40° angle lens with the same wd would allow for a FOV of $2 \cdot sin(20°) \cdot 912\text{mm} = 624\text{mm}$. The new spatial resolution of $384/624 = 1.62\text{mm}$ would still be more than five times higher than the benchmark nozzle density. The FOV can also be increased by using a lens with a longer wd.

- The spatial FWHM is at 1.2 pixels, which means that there will be some interaction between neighbours. In this integration, that should not affect the sorting due to the spatial resolution of the camera being more than 13 times that of the nozzle density. If the FOV is expanded significantly, this effect should be considered.

- The maximum spectral keystone of 0.21 shows that the spectrum read by one pixel would be almost the same as that read by another pixel under the same conditions looking at the same part of the sample. This maximum value is only at the edges of the detector, and the value goes down to 0.04 towards the middle.

- The spectral FWHM shows that there is a small interaction between neighbouring spectral bands. With a max value of 1.8 bands, will have some blurring between nearest neighbours.

- The spectral resolution of 5.4nm per band is high.

- The spatial smile has one "large" spike of 0.37 bands, which will be filtered out as a bad pixel, other than that spike the max is 0.08 bands. PCA and PLS-DA both needs a high SNR to work. Seeing that the max possible SNR is 1151 and that the saturation when doing the recordings has been at 80%, there is a good reason to believe that the SNR is high enough for PCA and PLS-DA.

- The second-order suppression test shows that the lower wavelengths do not affect the measurement of the longer ones.
- The dynamic range of 65536 gives a good colour depth, which gives high accuracy for the recorded spectra.
- The 12 bad pixels have been registered and interpolated away, which lowers the resolution at those points slightly, but they are evenly distributed, and far apart, it would have been worse if they had formed a cluster. They also only constitute $\frac{12}{384 \cdot 288} = 0.109\%$ of the pixels.

5.2 Samples

The samples were chosen as a proof of concept. They were therefore chosen on two criteria; availability and ease of classification, both with the camera and visually. The rock samples were available at a lab at NEO, and there is therefore a lot of information about them. The pieces of wood were found outside and polished a little to remove the aged layer. They fulfilled both criteria as they were easy to classify from each other, both for human eyes and with the hyperspectral camera. They were not that easy to separate from the background as will be discussed in workflow 1 and 2. This was probably due to them being so different, making the variance plot in the PCA only about the difference between them and not about the background.

5.3 Workflow 1

In workflow 1, the camera sends all the wavelengths to Breeze Runtime. Breeze Runtime removes 11 bands on each side, which is an automatic feature made because those bands usually are noisier than the rest.

For the recordings made for this workflow, an integration time of $5000\mu s$ was used as that gave a satisfactory saturation for the white reference. It was also tested that saturation did not go up to max at the most reflective samples, which are the wood samples.

Figure 4.2 is a pseudo RGB image used to visualize the image. The images can give some idea about how the samples look in the SWIR-region, but they use only 3 of 288 wavelengths, so it is not wise to conclude much from them. The wood pieces are bright and look almost saturated, while the rock pieces are quite dark. The colour depth of the wood pieces is, therefore, going to be deeper than for the rock pieces.

Separation

The creation of a separation model is shown in figure 4.3. The rocks have a homogeneous red colour, which points to the variance in the rocks being small. There are some edge effects on especially one of the rocks, but that is not a large enough piece to change the classification of the sample. The wood pieces are more inhomogeneous than they looked in the pseudo-RGB image, with a hue going from dark blue to green and yellow. This points to them being spread out in the variance plot, but at a distance from the rocks. This corresponds well to the variance plot, which has a tight cluster on the right corresponding to the rocks, and a wide blue arc on the left that corresponds to the wood pieces.

There are red dots scattered around the background; these are parts of the background being included in the separation model.

Figure 4.4 (a) shows a histogram based on the separation from figure 4.3. The vertical red line is used to choose how different a pixel can be from the model and still be included as a sample. A perfect separation model is only possible if the black bars can be separated from the grey bars by the red line. In this case, a compromise had to be made where grey bars where included, which leads to background pixels being included in the model. These pixels will come back as noise when sorting. It is countered by using a reasonable minimum object size, which excludes all the pixels not existing in big enough clusters.

Classification

The PLS-DA model is shown in figure 4.5. A separation between wood and rock using only the first dimension is possible, as a straight vertical line can be drawn that separates the blue and the green dots. A separation using only the second dimension is not possible as the wood and rock samples are overlapping in that direction. The model was created using both components. The first two components describe $91.4 + 7.98 = 99.38\%$ of the PLS-DA relationship between the wood and rock. Going down to one component could be considered for future application, as this would lower the computational intensity of creating the model.

Figure 4.6 shows the linear model made on the first two components of the PLS-DA-model and the corresponding R^2 and RMSE values. The values are $R^2 = 0.9976$ and $RMSE = 0.0216$. It must be noted that the model is now tested on the same pieces as it was trained with. The test does therefore not prove that the model can classify wood from rock, but that it can classify exactly these pieces of wood from exactly these pieces of rock. That is fine for this application since the model is going to be used to sort these pieces of wood, from these pieces of rock. A paper analysing the hardness of Maize kernels with hyperspectral imaging[32] also uses $RMSE$ to describe their PLS-DA-model. The lowest $RMSE$ value they achieve is 0.18, which is eight times more than the $RMSE$ of workflow 1. The linear separation of workflow 1 is, therefore, assumed to be solid.

Workflow

The combined workflow can be seen in figure 4.7. The measurements are first put through the sample model, which finds the objects. The objects are then sent to the WoodOrRock classification model, which splits the objects into the two groups, rock and wood. The orange boxes coming from the object box shows that the information about X and Width are sent from Breeze. X is the leftmost pixel at the left border of an object, and Width is the across track width of the objects.

The separation and classification were employed live on the same pieces of wood and rock as they were trained with. Figure 4.8 shows this. The blue pixels describe pixels classified as wood, while the green pixels are the pixels classified as rock. The white Xs and corners show where objects have been found. The rocks are well defined with almost no missing pixels. The wood pieces are identified properly, but there are many missing pixels inside. The background has many green pixels, especially on the bottom border of the sample mover, where there is almost a continuous line of green pixels. The separation model in the figure identifies all the samples, and the noise has no consequence on that decision making. The classification model does well as it does not mistake any parts of the wood pieces to be rock or vice versa.

5.4 Workflow 2

Workflow 2 limits the wavelengths recorded by the camera based on information gathered while making workflow 1. In figure 4.9, the most important wavelengths to the PLS-DA-model in workflow 1 are identified and shown. A new PCA-model was created with those wavelengths, but it did not give good separation between rock samples and background. Looking at the background spectra in figure 4.10 this spectra is very flat and slowly declining. Looking back at the outlined rock spectra in figure 4.9, one can see those also being quite flat and slanting downwards in the same manner as the background spectra.

The wavelengths were chosen manually to increase the separability. They were chosen based on two parameters; they should show visual characteristics in both the rock and wood spectra, and they should have a high importance number from PLS-DA. The wavelengths chosen are shown in figure 4.11 and as can be seen, they cover more of the peaks and features of the rock spectra. The separation improved, and it was decided to use those 56 wavelengths as the basis of workflow 2.

An integration time of $3500\mu s$ was used for the recordings in this workflow. That gave a saturation of 80% for the white reference. It was also tested at the wood samples as they were the most reflective samples, and it did not go into max saturation there.

Separation

The creation of a separation model is shown in figure 4.12. The rocks have a homogeneous blue colour, which points to the variance in the rocks being small. The wood pieces are somewhat inhomogeneous, going from yellow to red, this colour gradient points towards the dots being spread out in the scatter plot, but far away from the rocks. Breeze puts an orange colour over the gradient to represent that they are selected in the scatter plot. There are blue dots scattered around the background. These are parts of the background being included in the separation model. They share their colour with the rocks, and will probably be classified as rocks in the PLS-DA-model.

Figure 4.13 (a) shows a histogram based on the separation from figure 4.12. The model is not optimal here. There should ideally be a place to put the red line where the vertical bars are close to zero in height. In this model, grey bars had to be included to include as much of the samples as possible. This compromise had to be made to ensure that large enough parts of the samples are included. Quite a lot of the background pixels spread across the image are included in this model, which can be considered to be noise. They could get identified as samples if the clusters of noise get more extensive than the minimum object size of 600 pixels. On the right side of the scan, an almost continuous line of pixels is included. If this happens during a scan, clusters larger than 600 might form. That is the edge of the sample mover, however, and this edge will not be there in the final integration.

Classification

Figure 4.14 shows how the rocks and wood pieces are spreads out in a 2D space made out of the first and second PLS-DA-component. t[1] describing 97.2% of the covariance/variance is enough to linearly separate rock from wood. However, the model that was created uses two components. The first two components describe $97.2 + 2.22 = 99.42\%$ of the PLS-DA relationship between the wood and rock.

Figure 4.15 shows the linear model made on the first two components of the PLS-DA-model and the corresponding R^2 and $RMSE$ values. The values are $R^2 = 0.9997$ and $RMSE = 0.0091$.

The values from workflow 1 are $R^2 = 0.9976$ and $RMSE = 0.0216$. The new R^2 value is eight times closer to 1 than for Workflow 1, and the RMS error is 2.37 times smaller. The improvement could be due to the wavelengths being chosen specifically to increase the quality of classification. The wavelengths that describe similarities are therefore removed, keeping only the dissimilar ones. This kind of selection of the most important wavelengths is advised against in the paper [33, p. 224] as it can lead to a model that shows fake separation, even in the test set. If the goal had been to make a general model, a lot more care would have to be taken to ensure a diverse training set, and a separate test set to test for an overfitted model. This classification was done purely to have samples to test the system with, and the classification works well on the samples it was trained for.

Workflow

The combined workflow can be seen in figure 4.16. The X and Width is essential information for the FPGA so that it knows which valves to activate. Some non-linearity can occur in the distribution of spatial pixels on the sample mover. A spatial model should have been made relating each pixel with a spot on the sample mover. X and Width could then have been put through that spatial model to ensure that they are linearly behaving measurements.

Workflow 2 was employed live on the pieces of wood and rock it was trained with. Figure 4.17 shows this test. The rocks are identified properly. The wood pieces are also identified properly, but there are many missing pixels inside. The background has many green pixels, especially on the bottom border of the sample mover, where there is almost a continuous line of green pixels. It seems from the figure that the separation model finds the correct samples, but also misidentifies the edge to contain several samples.

5.5 Comparing workflow 1 and 2

- The two workflows where made with a different number of wavelengths used.
- Workflow 2 has a higher percentage in t[1] than workflow 1. This could mean that choosing the wavelengths based on properties in the samples, makes the variance describe those properties better.
- The scatter plot in workflow 1 is continuous, its hard to see where the dots representing the rocks ends and the ones representing the wood pieces begin. In workflow 2 this is clear with a large gap between them.
- Workflow 1 is less noisy when it comes to the background being included as a sample. This makes sense as the wavelengths in workflow 2 where chosen to maximize the quality of the classification, with some changes made based on visual interpretation to increase the separation.
- The wavelengths for workflow 2 would probably have been chosen better if a PLS-DA-model where made that classifies rock, wood and background from each other. That model would then have information about the background spectra. The most important wavelengths for that three way classification would probably be the best ones for separating the samples from the background and classifying the samples from each other. The ranking of wavelengths given there would therefore be more representative of the needs of the system than the ranking given by PLS-DA in this thesis. This could decrease the need for manual adjustment based on visual interpretation.

5.6 SynchSoftware

SynchSoftware receives object information from Breeze Runtime, collects them into frames and sends them to the FPGA. The SynchSoftware GUI allows the user to manage the camera, Breeze Runtime and Frequency Divider settings.

5.6.1 Max LPS

The author noticed that the software stops working properly when LPS exceeded 125Hz. This was investigated in the log file, and the speed of predictionLine processing was found to be the culprit. When LPS gets to high, the predictionLines are not processed fast enough. PredictionLines are read concurrently, this can create problems if the processing of a predictionLine takes longer than the period between predictionLines. When writing SynchSoftware, the author decided to focus on the code documenting its performance, rather than performing fast. There is therefore a logging function within predictionLines which writes status and info to a log file. This is a very time consuming task compared to the rather simple logic being done on each predictionLine.

The effect of a long processing time for the predictionLines can especially be seen in figure 4.22 and somewhat in figure 4.21. In figure 4.22 a massive delay between CaptureTime and ReceivedBySynch-Software is developed as the number of unprocessed predictionLines increases. In figure 4.21 there are two quick periods where the number of unprocessed predictionLines increases, probably due to some other process using more resources for some moments, and then the backlog is slowly taken care of until the two lines are indistinguishable again. In figure 4.20 no such deviation can be seen, which suggests that a LPS of 100Hz works well, while already at the 125Hz used in figure 4.21 delay starts to build up.

This delay creates two main issues:

1. The timing of when predictionLine is received is used to coordinate the creation of frames in SynchSoftware. When the predictionLines are delayed, the frames will be equally delayed.

2. A dictionary containing the timestamp and linenumber of each predictionLine is updated when receiving a predictionLine. This is used for finding the linenumber of the first and last line of each predictionObject. If the dictionary is not updated and the objects come before the predictionObjects, this will throw an error, and the predictionObject will not be sent.

How can this be fixed? In a new update of Breeze Runtime, Prediktera have now included linenumber for the first and last pixel of the object with the information sent with the event stream. That makse the predictionLineDictionary redundant, as the information does not need to be saved from predictionLine and used with predictionObject. So that solves problem 2.

Problem 1 cannot be completely solved without changing to a system where Breeze Runtime communicates directly with the FPGA. The LPS, at which this starts to occur, can be improved. The processing time of each predictionLine will be decreased by removing the logging. Saving to the shared dictionary can also be removed if using the new update of Breeze, this will also decrease the processing time. These two steps will probably make the system capable of handling much higher LPS. When removing the log functionality, an alternative method for checking for delays must be found. One solution is that the FPGA checks for delays in when the frames are sent. This will match the cumulative delay from the prediction lines.

5.6.2 Max speed of conveyor belt

The current max LPS of 125Hz corresponds to the max speed calculated in (5.1) when using square pixels.

$$\begin{aligned} v &= \frac{LPS \cdot FOV}{p} \\ &= \frac{125\text{Hz} \cdot 0.254\text{m}}{384\text{pixels}} \\ &= 0.083\text{ms}^{-1} \end{aligned} \tag{5.1}$$

The FPGA has a goal of at least running the conveyor belt at 1.7ms^{-1}. Which is 20.5 times faster than the current max speed with square pixels. Several steps can be taken to increase the max speed:

- Use rectangular pixels
- Increase FOV
- Increase max LPS

The spatial width and length of each pixel is currently as calculated in (5.2). Sub mm resolution might not be necessary. The speed of the conveyor belt could be increased by allowing the pixels to become rectangles. The rectangles could be scaled back in the software.

$$\begin{aligned} \frac{FOV}{p} &= \\ &= \frac{254\text{mm}}{384\text{pixels}} \\ &= 0.668\text{mm/pixel} \end{aligned} \tag{5.2}$$

When increasing FOV, the pixel width in the across track direction gets wider. The conveyor belt speed can then be increased while keeping the pixels square. If i.e. a 40° lens had been used, FOV would increase to 0.664m(5.3).

$$\begin{aligned} FOV &= 2 \cdot wd \tan\theta/2 \\ &= 2 \cdot 0.912\text{m} \tan 40^\circ/2 \\ &= 0.664\text{m} \end{aligned} \tag{5.3}$$

Which gives a pixel length of $0.664/384 = 1.7\text{mm/pixel}$, and a speed of $1.7\text{mm} \cdot 125\text{Hz} = 0.213\text{ms}^{-1}$.

Increasing LPS allows the speed to be increased while keeping the along track resolution. The max speed that can be achieved with square pixels, without considering the limitations of SynchSoftware and using Workflow 2 and a 40° lens is given by (5.4). LPS_{max} is calculated from using less than one fifth of the wavelengths, which allows the camera to run at more than five times the regular max LPS.

$$\begin{aligned} v_{max} &= \frac{LPS_{max} \cdot FOV_{40^\circ}}{p} \\ &= \frac{2000 \cdot 0.664}{384} \\ &= 3.458\text{ms}^{-1} \end{aligned} \tag{5.4}$$

5.6.3 Frame size

The current solution is that a frame is finished when the predictionLine corresponding to the end of the current frame + max object size arrives. A problem is that there is a delay between the arrival of objects compared to predictionLines. This makes makes the real max size for objects less than max object size. The author thought of two alternative methods for creating frames;

The first was to set a certain time period as the length of each frame, which would have the advantage of being a fixed amount of time. The size of a frame would then not be guaranteed to correspond with a set number of lines, especially when delays in the processing occurs. It would not solve the issue of objects using time before arriving to SynchSoftware, it would just make the issue relative to time, and not to lines.

The second is to wait for the bottom line of an arrived object to be larger than the max object size. This would ensure that all the objects that belong in a frame would be put in that frame, however, it would make the frames unpredictable. If a new object does not arrive with a bottom line larger than max object size, then the frame would not be sent. A timeout could be set to solve this, but that would lead to inconsistent behaviour.

The objects that arrived to late to be put in the correct frame was omitted, instead of being put in the next frame. The FPGA cannot handle objects with a negative position, which they would have in that situation. Figure 4.19 shows how some object gets omitted when max object size is set to 50, while the mean object size is around 60. This looks to roughly follow the probability calculated in (3.3).

All the objects that where larger than max object should probably be omitted, that would make for a more consistent and predictable sorting not relaying on statistics. That is a trivial task to implement.

5.6.4 Frequency divider

The frequency divider sends out a trigger pulse to the camera and a frequency divided event trigger pulse to the FPGA. The event trigger pulse being frequency divided makes it essential that the trigger pulse comes with a constant period between each rising flank. SynchSoftware assumes that the lines inside each frame are evenly spaced, so any deviation would affect how accurate the objects are placed within each frame. That would again affect how accurate the objects are blown by the valves.

In table 4.2 the accuracy of the trigger pulse and event trigger pulse was measured. The trigger pulse were measured at the periods 5.064ms and 1ms, with the event trigger pulse being frequency divided by 128 to respectively 648.2ms and 128ms. Unfortunately the available oscilloscope could only measure the min and max deviation down to 1/1000th of the value. However, this still provides the information that all the trigger pulses during the 961 cycles deviated less than 1μs. In the future the conveyor belt might be driven at 3ms^{-1}. Even at this high speed the 1μs would only make the placement of the object deviate with: $3 \cdot 1 \cdot 10^{-6} = 3\mu$m. The smallest samples considered to be used are 3cm, which is

10^5 times larger than the deviation.

5.6.5 Camera Link timestamp

The stability of the Camera Link timestamp was tested by running the camera for 10 minutes in a free-run mode with internal triggering. Figure 4.18 shows the results of this test. The top figure shows the difference between consecutive timestamps. For the timestamp to be accurate enough to be used as a reference each timestamp should correspond to when the line was scanned. At $LPS = 395$Hz, the time between each line scanned is 2.53μs. That corresponds to $0.253 \cdot 10^5[100ns]$ in the graph. If the timestamp had been accurate, the time diff should have made one line at that level. Instead there are five lines that looks like multiples of 0.253, and one near zero, plus a lot of noise. It seems like the frame grabber has delays consisting of a multiple number of the base frequency, and then works through the frames it missed really fast. The biggest Time diff in this graph is at about $2.5 \cdot 10^5[100ns]$ corresponding to $2.5 \cdot 10^5 \cdot 100\text{ns} = 25\text{ms}$. That is 10 times more than the period between consecutive lines scanned.

In the bottom graph, that should have been linear if the signal was stable, there are suddenly huge gaps of time between consecutive lines. The biggest one is at approximately $0.01 \cdot 10^9[100ns] = 1\text{s}$. If the frame grabber has a delay of a full second, more problems than an inaccurate timestamp will take place. This should be investigated further.

Chapter 6

Conclusion

The goal of this thesis is to prepare a classification module that can be integrated with any sorting machine. This goal is achieved with a hyperspectral camera, Breeze Runtime, and a Frequency Divider. Two classification workflows are described for the classification module. They can classify three pieces of wood from three pieces of rock. Workflow 1 uses the full range of wavelengths the camera can supply. Workflow 1 had a successful separation and the classification model has $R^2 = 0.9976$ and $RMSE = 0.0216$. Workflow 2 uses only 56 out of 288 wavelengths, which allows the camera to increase its max frequency to 2000Hz. The wavelengths for Workflow 2 are picked by hand to both increase the separability to the background and the classification between wood and rock. The separation model in Workflow 2 is noisier than the one in Workflow 1. The classification model is better for workflow 2, as $R^2 = 0.9997$ which is closer to 1, and $RMSE = 0.0091$ which is smaller.

SynchSoftware was developed as part of this thesis. It coordinates and communicates between the classification module and the FPGA. It has a GUI that gives the user control over all the subsystems, and it facilitates the communication between the classification software and the sorting machine. During the test of a specific integration performed in this thesis, SynchSoftware is found to be limited to frequencies below 125Hz. That limitation is due to SynchSoftware using most of its time on logging. The timing of the system relies on the stability of the Frequency Divider. It was found to be stable with a deviation of less than one thousandth of the trigger signal period.

6.1 Further work

The FOV can be expanded. This can be deduced from the current FOV being narrow compared to regular conveyor belts, as well as the concentration of pixels being much higher than the concentration of nozzles. The max LPS can be increased. SynchSoftware currently has a limit of approximately 125Hz, which can be increased by removing logging functionality and updating to a new version of Breeze that includes linenumber.

In future cases, it could be an idea to make a three-class PLS-DA-model with the background as one of the samples and use that model's interpretation of which wavelengths are the most important. The current system of estimating which wavelengths to use by looking at which wavelengths have the most effect on the PLS-DA-model does not ensure a good separation from the background, due to there being no information about the background.

Chapter 7

Bibliography

[1] HySpex, Hyperspectral Imaging. [Online]. Available: https://www.hyspex.com/hyperspectral-imaging/

[2] F. Pettersson, "Quantification of APIs using PLS."

[3] A. Vidman, "Introduction To Hyperspectral Imaging Analysis and the Evince and Breeze Software."

[4] S. Wold, M. Sjöström, and L. Eriksson, "PLS-regression: A basic tool of chemometrics," vol. 58, no. 2, pp. 109–130. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0169743901001551

[5] richlander. .NET Core intro and overview. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/core/introduction

[6] D. M. Hawkins, "The Problem of Overfitting," *Journal of Chemical Information and Computer Sciences*, vol. 44, no. 1, pp. 1–12, Jan. 2004.

[7] Serilog — simple .NET logging with fully-structured events. [Online]. Available: https://serilog.net/

[8] S. Hedberg. Prediktera – Hyperspectral Imaging Software. [Online]. Available: https://prediktera.com/

[9] R. Moscetti, W. Saeys, J. C. Keresztes, M. Goodarzi, M. Cecchini, M. Danilo, and R. Massantini, "Hazelnut Quality Sorting Using High Dynamic Range Short-Wave Infrared Hyperspectral Imaging," vol. 8, no. 7, pp. 1593–1604. [Online]. Available: https://doi.org/10.1007/s11947-015-1503-2

[10] W. Wang and C. Li, "A multimodal machine vision system for quality inspection of onions," vol. 166, pp. 291–301. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0260877415002824

[11] O. Kleynen, V. Leemans, and M. F. Destain, "Selection of the most efficient wavelength bands for 'Jonagold' apple sorting," vol. 30, no. 3, pp. 221–232. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925521403001121

[12] L. A. Paluchowski, E. Misimi, L. Grimsmo, and L. L. Randeberg, "Towards automated sorting of Atlantic cod (Gadus morhua) roe, milt, and liver – Spectral characterization and classification using visible and near-infrared hyperspectral imaging," vol. 62, pp. 337–345. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0956713515302772

[13] P. Tatzer, T. Panner, M. Wolf, and G. Traxler, "Inline sorting with hyperspectral imaging in an industrial environment," in *Real-Time Imaging IX*, vol. 5671. International Society for Optics and Photonics, pp. 162–173. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/5671/0000/Inline-sorting-with-hyperspectral-imaging-in-an-industrial-environment/10.1117/12.601192.short

[14] A. C. Karaca, A. Ertürk, M. K. Güllü, M. Elmas, and S. Ertürk, "Automatic waste sorting using shortwave infrared hyperspectral imaging system," in *2013 5th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, pp. 1–4.

[15] Archiwa: Case studies -. [Online]. Available: http://www.comex-group.com/k/case-studies

[16] C.-K. Liang, L.-W. Chang, and H. H. Chen, "Analysis and Compensation of Rolling Shutter Effect," vol. 17, no. 8, pp. 1323–1330.

[17] *Infrared Spectroscopy for Food Quality Analysis and Control*. Elsevier. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/B9780123741363X00016

[18] T. Skauli and H. E. Torkildsen, "Simplified measurement of point spread functions of hyperspectral cameras for assessment of spatial coregistration," in *Algorithms, Technologies, and Applications for Multispectral and Hyperspectral Imagery XXV*, D. W. Messinger and M. Velez-Reyes, Eds. SPIE, p. 13. [Online]. Available: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/10986/2520219/Simplified-measurement-of-point-spread-functions-of-hyperspectral-cameras-for/10.1117/12.2520219.full

[19] J. Workman, Jerry and L. Weyer, *Practical Guide to Interpretive Near-Infrared Spectroscopy*, 0th ed. CRC Press. [Online]. Available: https://www.taylorfrancis.com/books/9781420018318

[20] F. Pettersson, "A multivariate approach to computational molecular biology." [Online]. Available: http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-609

[21] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," vol. 2, no. 1, pp. 37–52. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0169743987800849

[22] T. Ilani, I. Herrmann, A. Karnieli, and G. Arye, "Characterization of the biosolids composting process by hyperspectral analysis," vol. 48, pp. 106–114. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0956053X15302257

[23] S. P. Gundupalli, S. Hait, and A. Thakur, "A review on automated sorting of source-separated municipal solid waste for recycling," vol. 60, pp. 56–74. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0956053X16305189

[24] C. Apetrei, I. M. Apetrei, S. Villanueva, J. A. de Saja, F. Gutierrez-Rosales, and M. L. Rodriguez-Mendez, "Combination of an e-nose, an e-tongue and an e-eye for the characterisation of olive oils with different degree of bitterness," vol. 663, no. 1, pp. 91–97. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0003267010000942

[25] K. P. F.R.S, "LIII. On lines and planes of closest fit to systems of points in space," vol. 2, no. 11, pp. 559–572. [Online]. Available: https://doi.org/10.1080/14786440109462720

[26] Thraka. Getting Started - WPF. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/framework/wpf/getting-started/

[27] O. Jonsson and A. Vidman. Breeze Runtime, Developers Reference Guide. [Online]. Available: https://www.prediktera.com/download/pdf/Prediktera_Breeze_Runtime_Documentation.pdf

[28] "NI 9401 Datasheet - National Instruments," p. 10. [Online]. Available: http://www.ni.com/pdf/manuals/374068a_02.pdf

[29] "NI 9151 Datasheet.pdf." [Online]. Available: http://www.ni.com/pdf/manuals/373560b.pdf

[30] HySpex, SWIR-384. [Online]. Available: https://www.hyspex.com/hyspex-products/hyspex-classic/hyspex-swir-384/

[31] USB oscilloscopes & mixed signal oscilloscopes - picoscope 2000A. [Online]. Available: https://www.picotech.com/oscilloscope/2000/picoscope-2000-overview

[32] P. Williams, P. Geladi, G. Fox, and M. Manley, "Maize kernel hardness classification by near infrared (NIR) hyperspectral imaging and multivariate data analysis," vol. 653, no. 2, pp. 121–130. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000326700901191X

[33] R. G. Brereton and G. R. Lloyd, "Partial least squares discriminant analysis: Taking the magic away," vol. 28, no. 4, pp. 213–225. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/cem.2609

Appendix A

Appendix

A.1 Camera test report

"norsk elektro optikk"

HySpex SWIR-384

SN-**3135**

Hyperspectral Camera

Test Report

"norsk elektro optikk"

Test Report
HySpex SWIR-384, SN-3135

	NAME	DATE
Tests performed by	*Knut Ola Bjerke*	*21/12/16*
Approved by	*Ivar Baarstad*	*21/12/16*

norsk elektro optikk

Test Report
HySpex SWIR-384, SN-3135

"norsk elektro optikk"

Test Report
HySpex SWIR-384, SN-3135

1 Introduction

This document describes the tests performed on the hyperspectral camera to document that the instrument is properly assembled and adjusted and that the instrument works according to specifications before delivered to the end user.
Please note that the test results and procedures should not be disclosed or published without written consent from Norsk Elektro Optikk AS.

2 Instrument details

Customer: NEO DEMO
Instrument serial #: 2016 SN-3135 (SWIR-384)

3 Test procedures/results

3.1 Spatial resolution

3.1.1 Test procedure

The spatial resolution has been measured by positioning a broad band point source (size 25µm) at infinite object distance (simulated by means of a collimator) and the FWHM of the point source was plotted as a function of wavelength.

3.1.2 Test results

Point source measurements are shown in the following figures.

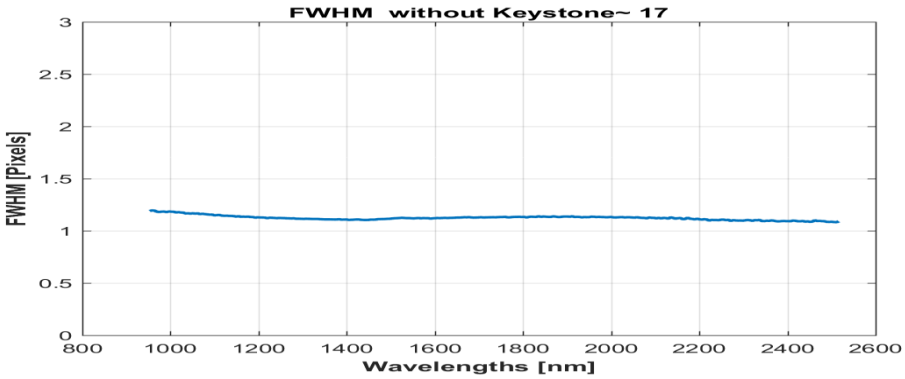


Figure 1. FWHM of point source as a function of wavelength for spatial pixel number ~18.

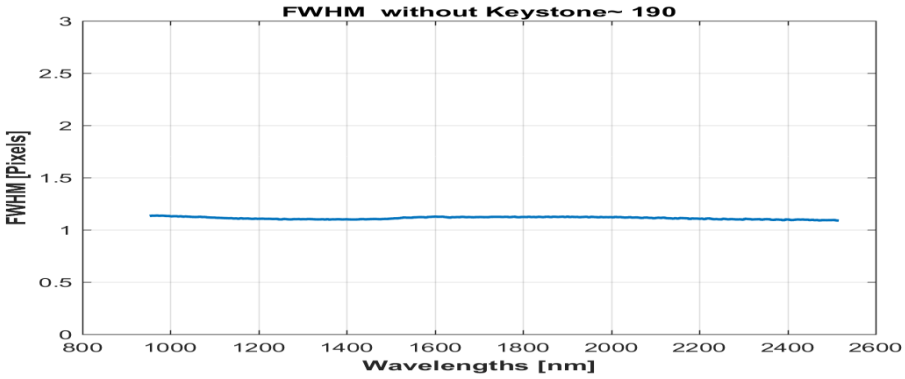


Figure 2. FWHM of point source as a function of wavelength for spatial pixel number ~190.

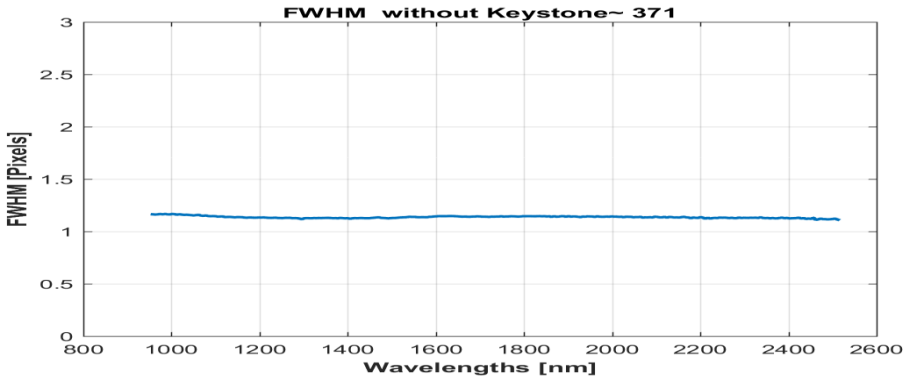


Figure 3. FWHM of point source as a function of wavelength for spatial pixel number ~370.

3.2 Spatial misregistration (keystone)

Spectral keystone effect is an optical distortion that causes deviations in the spatial position of a point source for different wavelengths, also called spatial misregistration. The performed tests are also used to check the alignment of the grating with the sensor/slit.

3.2.1 Test procedure

Same as point source measurements in 3.1.1, but center of gravity of point source (in spatial direction) is plotted as a function of wavelength.

3.2.2 Test results

The following figures show the test results for different positions (pixel numbers) in the FOV.

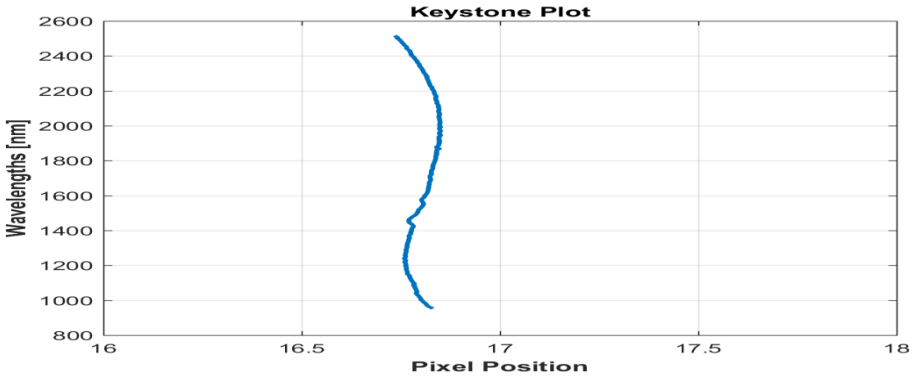


Figure 4. Position of center of gravity of a point source as a function of wavelength (band number).

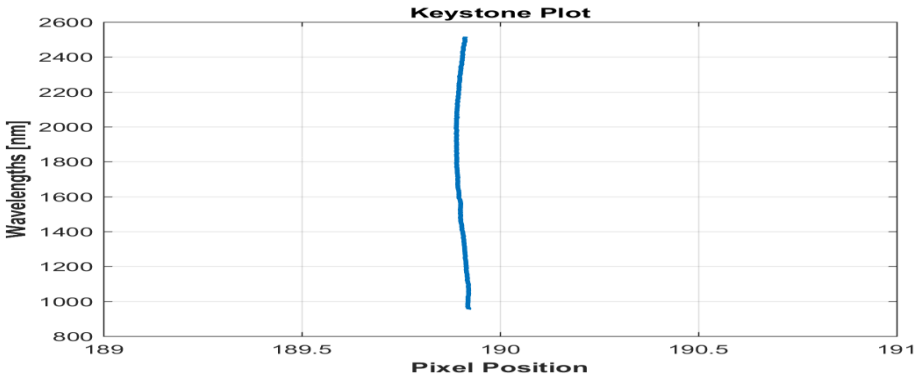


Figure 5. Position of center of gravity of a point source as a function of wavelength (band number).

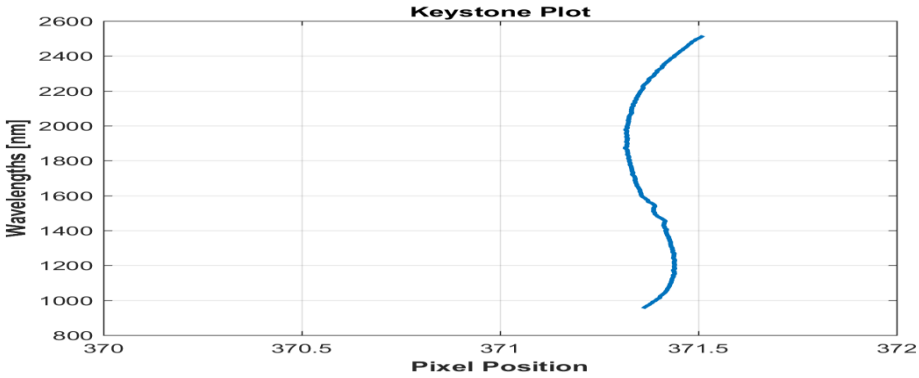


Figure 6. Position of center of gravity of a point source as a function of wavelength (band number).

The results show that the keystone effect is a small fraction of a pixel, and negligible for most practical purposes. Furthermore, the results show that the grating is very well aligned with the sensor and slit.

3.3 Spectral resolution

3.3.1 Test procedure

The spectral resolution, spectral sampling and smile effect has been measured by illuminating an integrating sphere with different narrowband light sources of wavelengths. The distribution of the light from these narrowband sources is used for the analysis in sections 3.3 and 3.4.

3.3.2 Test results

The results of these measurements are shown in the following figures.

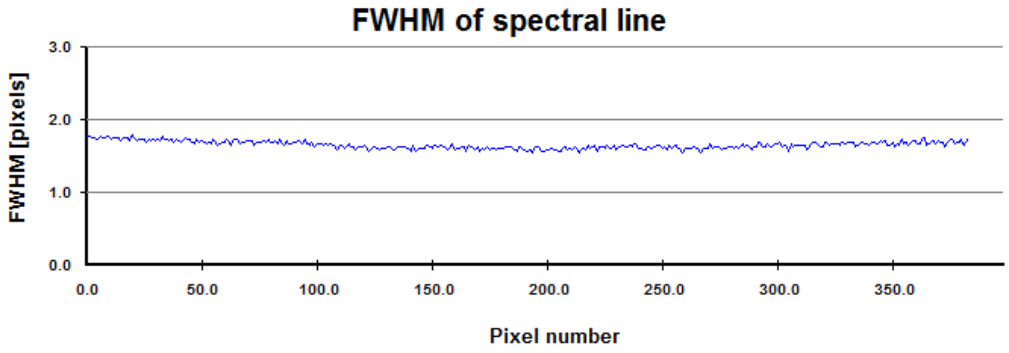


Figure 7. FWHM of 1014 nm HgAr line as function of spatial pixel (weak source).

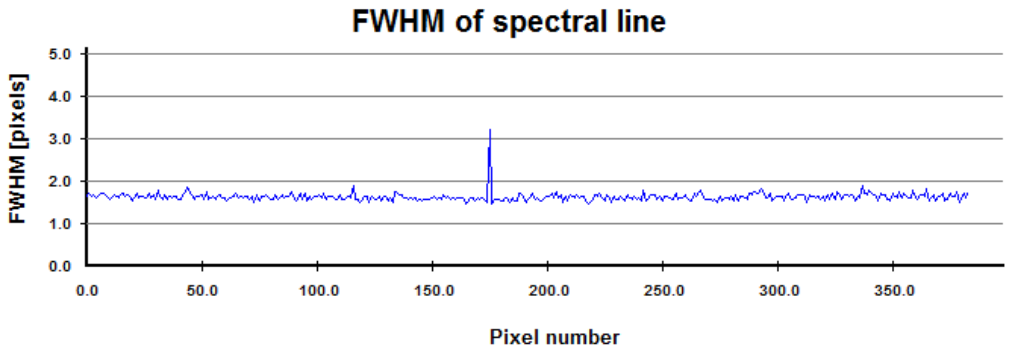


Figure 8. FWHM of 1442.7 nm Kr line as function of spatial pixel (very weak source).

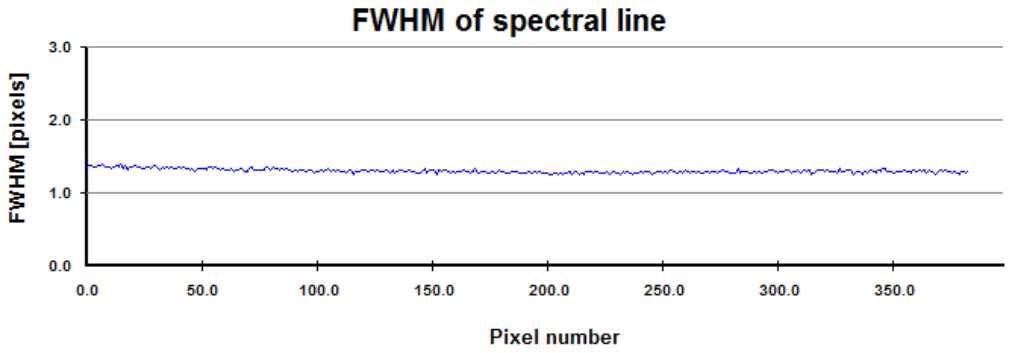


Figure 9. FWHM of 1694 Argon line as function of spatial pixel.

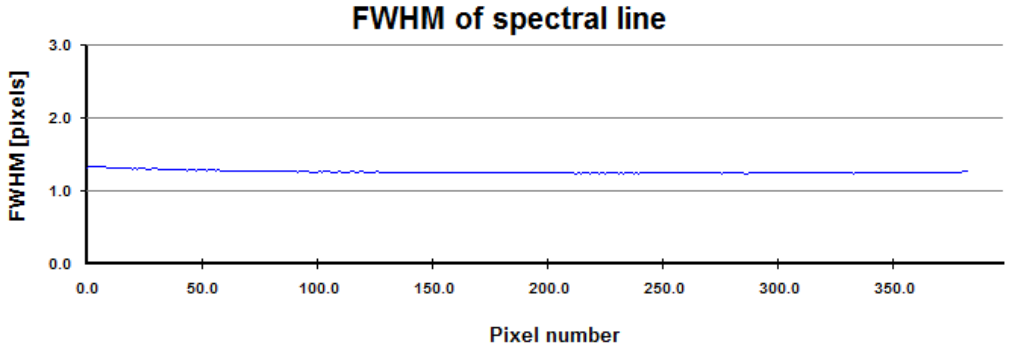


Figure 10. FWHM of 2026.2 nm Xe line as function of spatial pixel.

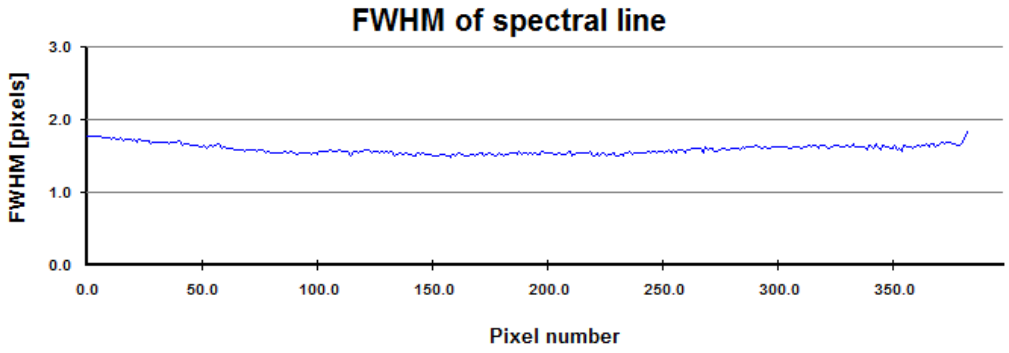


Figure 11. FWHM of 2482.4 nm Xe line as function of spatial pixel.

IMPORTANT:
Note that, when the spectral line hits the centre of a pixel row, then most of the illumination is contained within this row. If the spectral line illumination does not hit the centre of a row, the FWHM plots will appear wider, even though the image quality is very good. This is because the trapezoidal method is used to calculate the FWHM, and if the illumination is evenly distributed between two pixels, the result will be close to 2, even though the physical width of the line is only on the order of 1 pixel. The FWHM plots should therefore be studied together with the plots of spectral misregistration in the following section.

3.4 Spectral misregistration (smile)

Smile effect is an optical distortion that can be seen as a bending of a spectral line across the FOV. This effect is minimized in the HySpex optical design, and for most practical purposes negligible. The effect is characterized using the same measurements as described in section 3.3. It is measured by plotting the centre of gravity of a spectral line as a function of spatial pixel. This is also used to check the alignment of the sensor with the slit and the total spectral misregistration.

3.4.1 Test procedure

See section 3.3.1.

3.4.2 Test results

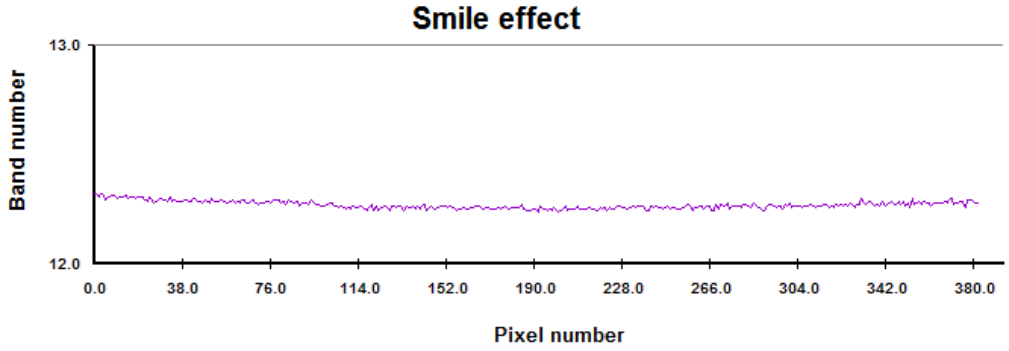


Figure 12. Center of gravity as a function of spatial pixel (smile effect) of 1014 nm HgAr line as function of spatial pixel (weak source).

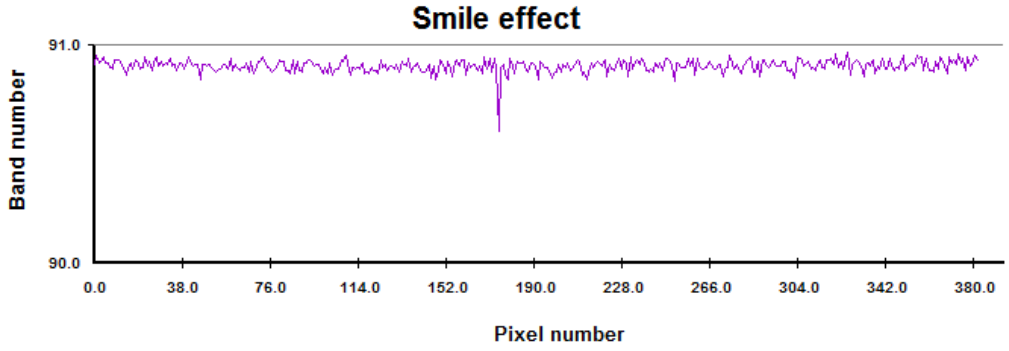


Figure 13. Center of gravity as a function of spatial pixel (smile effect) of 1442.7 nm Kr line as function of spatial pixel (weak source).

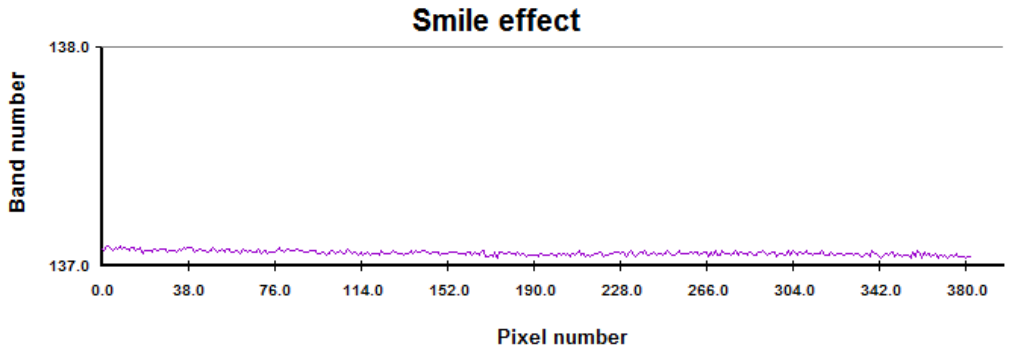


Figure 14. Center of gravity as a function of spatial pixel (smile effect) of 1694 Ar line as function of spatial pixel.

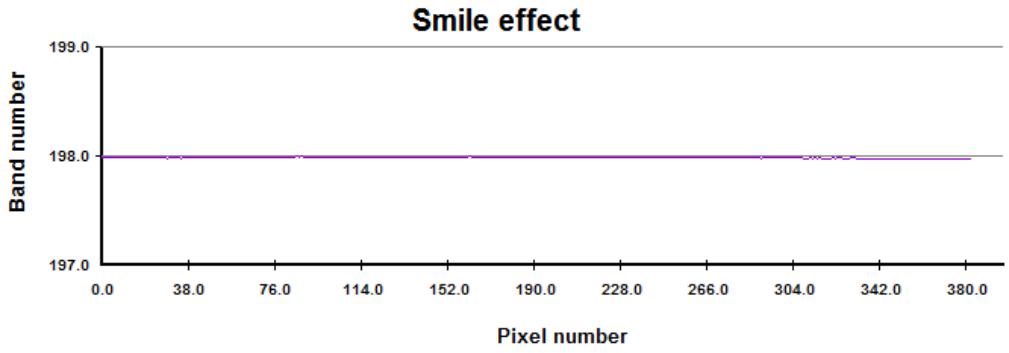


Figure 15. Center of gravity as a function of spatial pixel (smile effect) of 2026.2 nm Xe line as function of spatial pixel.

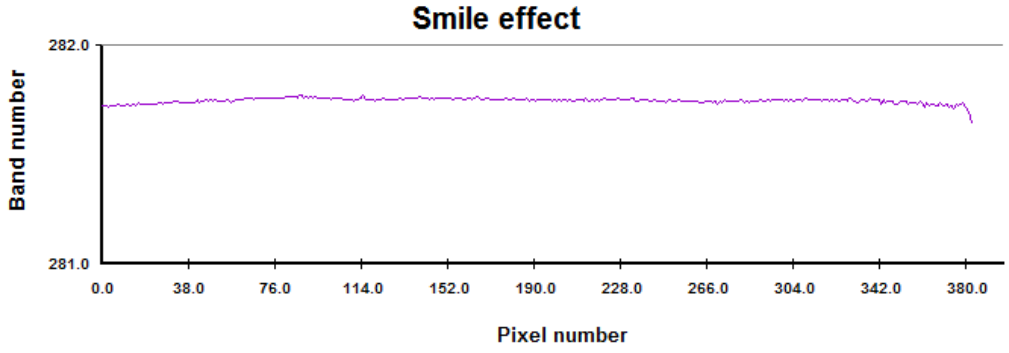


Figure 16. Center of gravity as a function of spatial pixel (smile effect) of 2482.4 nm Xe line as function of spatial pixel.

The graphs show that the smile effect is negligible for most practical purposes. The measurements also demonstrate that the slit and sensor are well aligned.

3.5 Responsivity matrix

The responsivity for each pixel on the 2D sensor array has been measured and scaled to produce calibrated images with pixel spectra in units of radiance (W/m^2 nm sr).

3.5.1 Measurement procedure

The instrument was pointed into a radiometric calibrated integrating sphere, illuminated with a broad band light source, resulting in a uniform scene of known radiance.

3.5.2 Measurement results

The results here present the overall signal level during the calibration measurements along the spectral and spatial dimensions. The complete responsivity matrix is included in the delivery (see .set file in the HySpex directory).

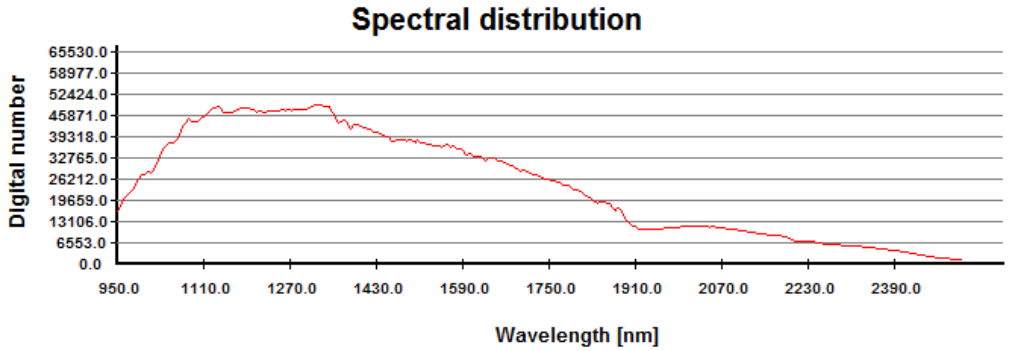


Figure 17. DN level as a function of wavelength for spatial pixel 160 during radiometric calibration measurements.

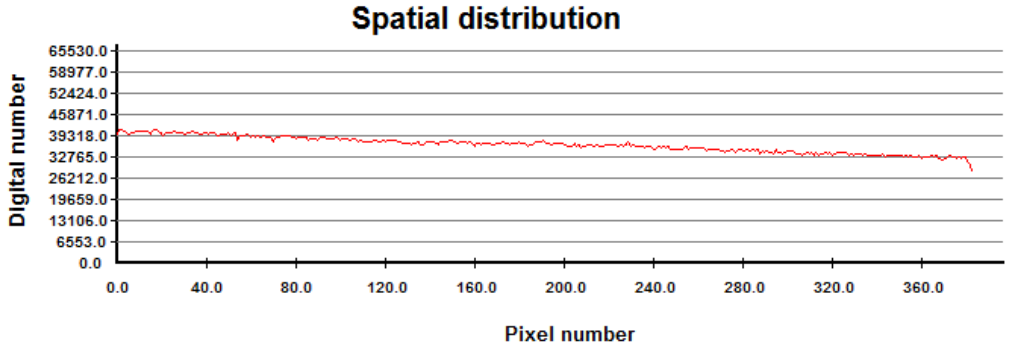


Figure 18. DN level as a function of spatial pixel for 1275 nm during radiometric calibration measurement.

3.6 Second Order Suppression

A general test of the Second order signal can be performed by checking the signal level in the blocking region of short wave pass 2460 filter that is mounted in the entrance aperture.

3.6.1 Test procedure

The instrument was pointed into an integrating sphere illuminated with a broad band light source. An SWP 1600 filter was mounted in the entrance aperture. The measurement of the signal level in the blocking region confirms that the Second order is blocked. It also makes sure that the order sorting filter is at correct position and the performance is similar from instrument to instrument.

3.6.2 Test results

The spectra recorded from the filter measurements are shown in the following figure.

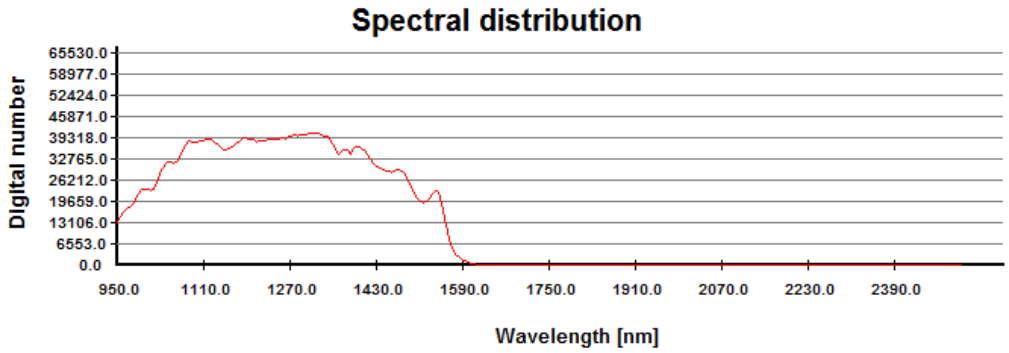


Figure 19. Signal level with short wave pass 1600 filter.

The graph demonstrates that there is no measurable second order signal reaching the sensor (would have been seen as a bump in the spectrum).

3.7 Signal/Noise

3.7.1 Test procedure

The instrument was pointed into an integrating sphere illuminated with a stable DC driven non-flicker broad band illumination, and the integration time was adjusted so that some pixels were saturated. The mean signal level vs the standard deviation was plotted for all pixels.

Additionally, the mean vs standard deviation was plotted for no signal, showing the noise floor of the sensor. Finally, the SNR as a function of wavelength was plotted for the light distribution measured in the integrating sphere.

The spectral distribution of the integrating sphere is as follows:

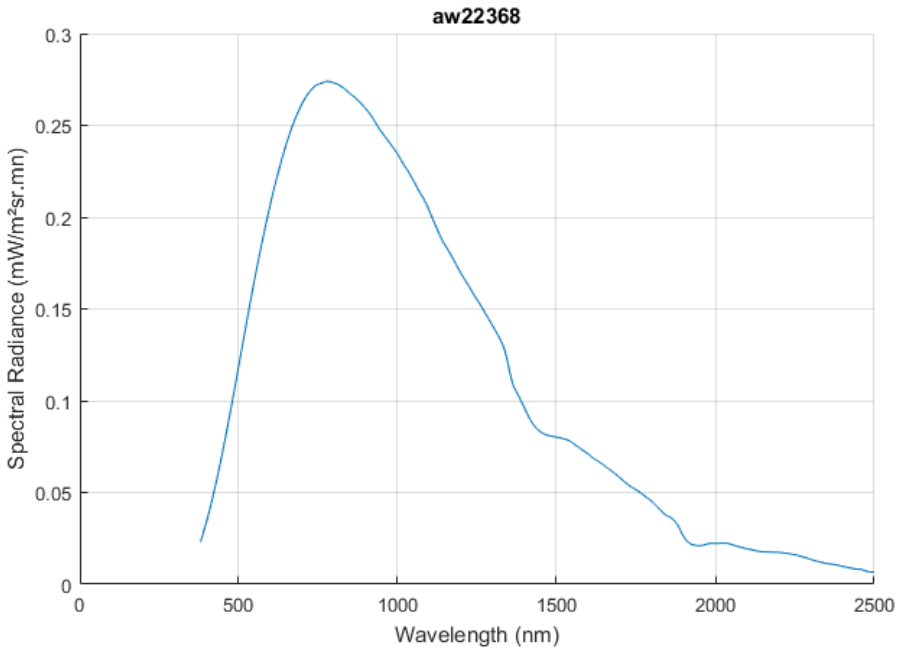


Figure 20. Radiance from integrating sphere.

3.7.2 Test results

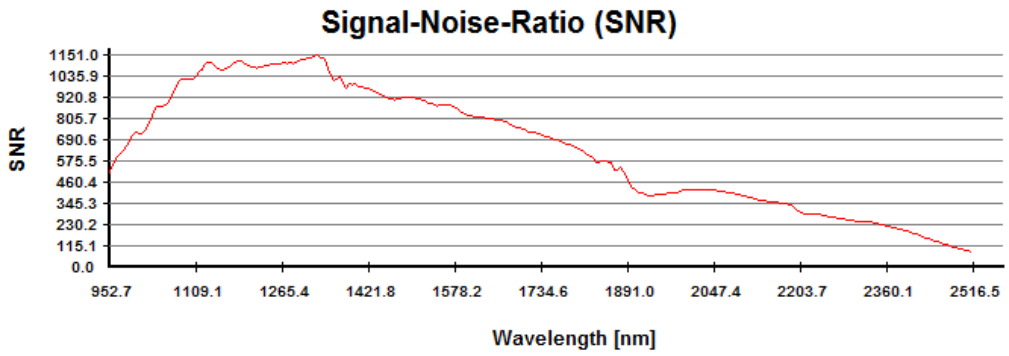


Figure 21. SNR as a function of wavelength for sphere spectrum (see Figure 20).

The real SNR for a given scene will of course depend on the radiance level.

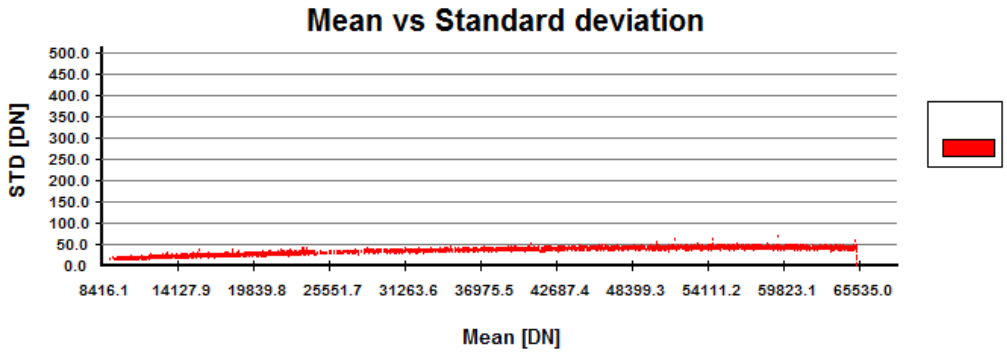


Figure 22. Standard deviation as a function of mean signal level with broad band illumination.

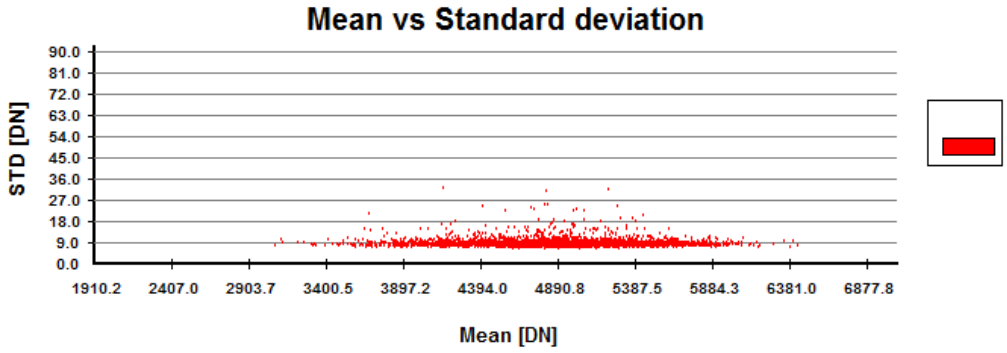


Figure 23. Standard deviation as a function of mean signal level, no illumination (100us integration time).

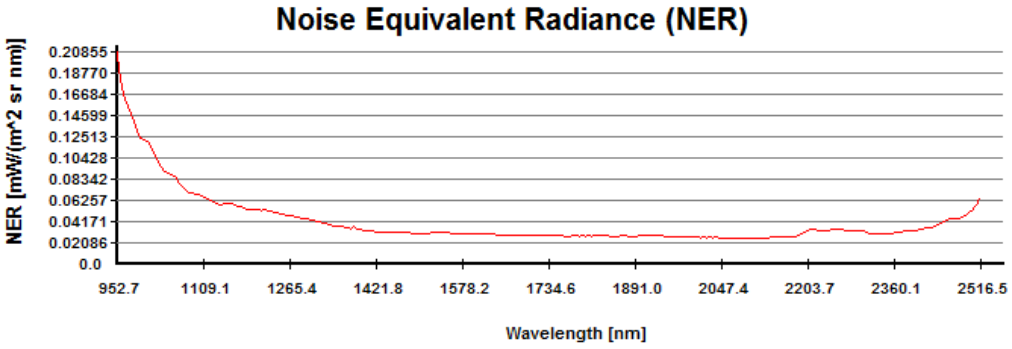


Figure 24. NER as a function of wavelength measured with 10ms integration time. Conversion from DN's to radiance is based on the sensor calibration values.

3.8 Shutter efficiency

The HySpex systems are equipped with an automatic electro-mechanic shutter which is used to acquire one background (dark offset) data set for each image acquisition, in order to make sure that a correct dark offset is always subtracted.

3.8.1 Test procedure

The system was pointed into an integrating sphere with broad band illumination. The integration time was adjusted to the saturation level and the shutter was closed. One data set was acquired while the entrance aperture was open towards the illumination from the sphere, and one data set was acquired with the exit port of the sphere blocked, thus stopping all light from entering the instrument. Then the "dark" measurement was subtracted from the "light" measurement and the resulting values are illustrated below.

3.8.2 Test results

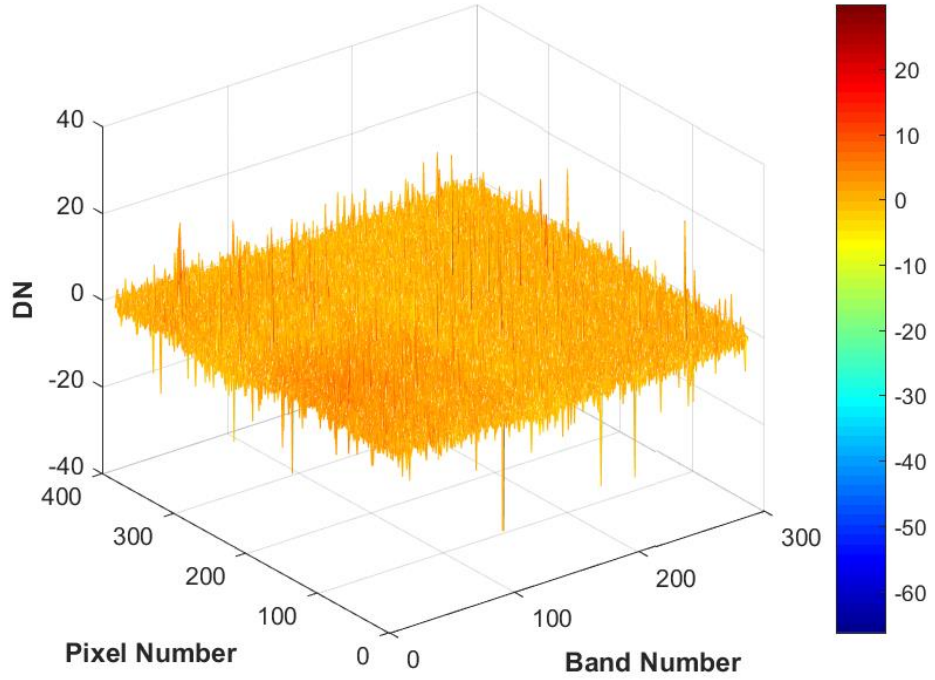


Figure 26. Result of shutter test.

The figure shows that no light is reaching the sensor when the shutter is closed.

3.9 Bad pixels

A detailed characterization of bad pixels on the sensor array has been performed, in order to classify dead pixels, hot pixels, noisy pixels and other deviating pixels. This characterization has been done by acquiring several sets of data under varying light conditions and performing a detailed analysis of these data sets. The bad pixel list is found is the settings file (.set) for the sensor, found in the HySpex GROUND directory on the DAU.
The bad pixel map is included below.

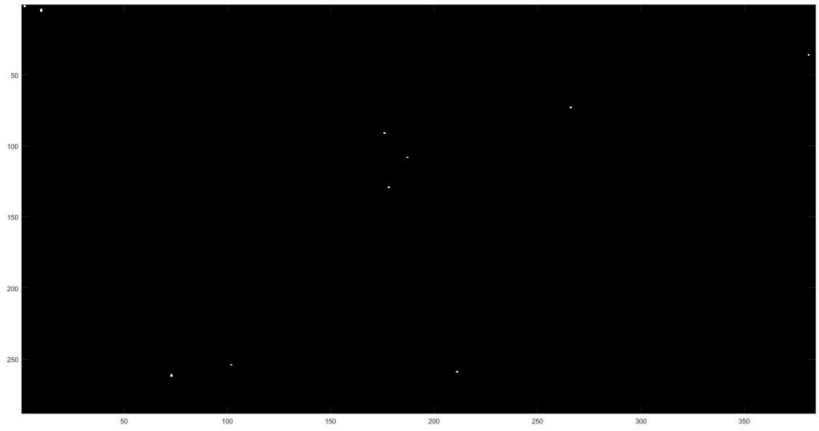


Figure 27. Bad pixel map.

3.10 Sensor model

The sensor model provides information about the angular field of view across the scan direction for all the spatial pixels, as well as the total field of view. This information is used as an input when georeferencing airborne data.

3.10.1 Test procedure

The camera was pointed at a broad band point source and the sensor was scanned with a high precision and high resolution rotation stage across the whole FOV of the sensor.

3.10.2 Test results

The sensor model test result for the sensor is shown in the following figure.

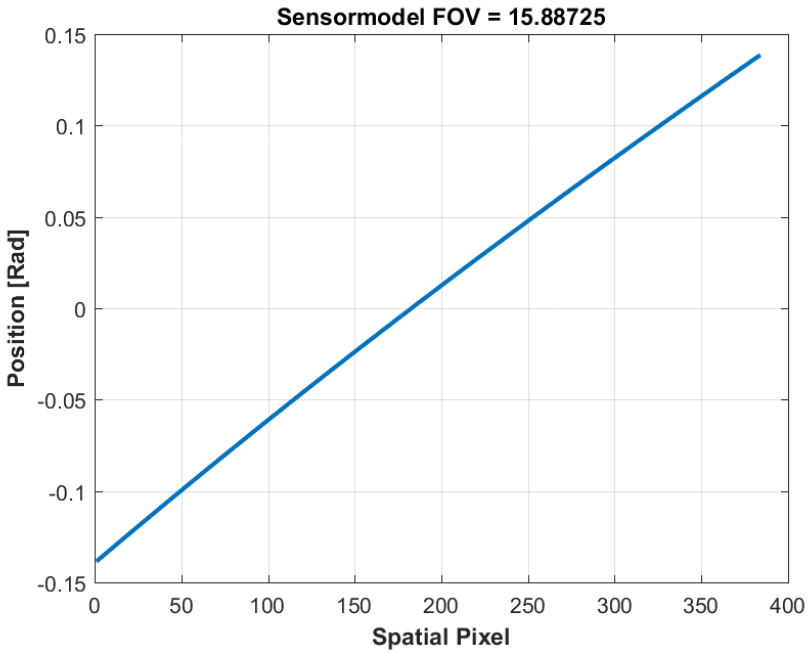


Figure 28. Sensor model without fore-optics.

The figures show the pointing angle of each pixel, normalized to have same pointing angle for the edge pixels at each side of the FOV (but with opposite signs). These results are also available as txt-files, normally stored on the desktop of the DAU.

4 Conclusion

All the main relevant characteristics of the HySpex imaging spectrometer system have been tested and characterized. The results are in line with the specifications.

Appendix A: Close-up lenses

Working distance with microscope, 30cm and 1m lens were tested.
The optimal working distance for the microscope is 28 mm, -for 30cm and 1m lenses 21,5cm and 91,2cm respectively.

Mørenskog, Erik Rundhovde

Real-time hyperspectral sorting

NTNU
Norwegian University of
Science and Technology