A Comparison of Airborne and Simulated EnMap Hyperspectral Imagery for Mapping Bedrock Classes in the Canadian Arctic

by

Roger MacLeod
B.A., Lakehead University, 1997

A Thesis Submitted in Partial Fulfillment
of the Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Geography

Supervisory Committee

A Comparison of Airborne and Simulated EnMap Hyperspectral Imagery for Mapping Bedrock Classes in the Canadian Arctic

by

Roger MacLeod
B.A., Lakehead University 1997

Supervisory Committee

Dr. K. O. Niemann, (Department of Geography)
Supervisor

Dr. J. R. Harris, (Geological Survey of Canada)
Additional Member

Abstract

Dr. K. O. Niemann, (Department of Geography)
Supervisor

Dr. J. R. Harris, (Geological Survey of Canada)
Additional Member

The upcoming launch of the German hyperspectral satellite: Environmental Mapping and Analysis Program (EnMAP) will provide potential for producing improved remotely sensed maps in areas of exposed bedrock in advance of Arctic geology programs. This study investigates the usefulness of this moderate resolution (30m) sensor for predictive lithological mapping using simulated imagery to classify a map area dominated by mafic and felsic volcanics and minor sedimentary and volcaniclastic rocks in the Hope Bay Greenstone Belt of the Northwest Territories. The assessment also included the classification of high resolution and fidelity airborne (ProSpecTIR–SPECIM Dual sensor) hyperspectral imagery for comparison to understand the impact of combined lower signal-to-noise ratio (SNR), and spectral and spatial resolutions associated with EnMap.

The performance of both sensors was assessed through statistical analysis of the classification results based on partial unmixing of the data as well as common geological band indices. The results obtained from these analyses were compared to a detailed published geological map of the study area.

Both sensors, the airborne ProSpecTIR–SPECIM and spaceborne EnMap, provided good results however despite the simulated EnMap data's lower resolution and SNR, the results showed it to have greater statistical accuracy and to be visually representative of the mapped geology. The results demonstrated that EnMap satellite hyperspectral technology is an effective tool for mapping lithology in the Canadian North. The discrimination of rock compositions was successful when their occurrences were spatially large and abundant; however, it was identified that spectral similarity between unit classes and spectral variability within classes are critical factors in mapping lithology.

Table of Contents

Supervisory Committee ii

Abstract iii

Table of Contents v

List of Figures ix

List of Tables xiv

List of Abbreviations xv

Chapter 1: Introduction 1

1.1 Background 1

1.2 Research Goals and Objectives 11

1.2 Thesis Structure 12

Chapter 2: Factors Influencing the Accuracy of Predicted Lithological Units Through the use of Hyperspectral Imagery 15

2.1 Introduction 15

2.2 Scene Specific Considerations 16

2.2.1 Scene Geometry 16

2.2.2 Atmospheric Considerations 23

2.2.3 Sub- And Pixel Level Landscape Spatial Structure 25

Spectral properties of rock minerals 25

Background and targeted spectral differences 25

Variance of targets and background 26

Landscape spatial structure patterns 27

2.3 Sensor Specific Considerations..... 27

2.3.1 Spatial Resolution 28

Sensor Design 28

Classification influences 29

2.3.2 Spectral resolution & wavelength coverage 31

2.3.3 Radiometric resolution..... 33

2.3.4 Image Quality..... 34

2.3.5 Sensor Type 36

2.3.6 Spectral and Spatial Distortions Keystone and Smile..... 38

2.6: Dual Spectrometer Co-Registration..... 39

2.3.8 Scene Size Capability 40

2.4 Processing Considerations 41

2.5 Conclusion 42

Chapter 3: Simulation of EnMAP Data through Upscaling of Airborne HSI Data..... 44

3.1 Introduction..... 44

3.2 Required Input Data..... 48

3.4 Spatial Convolution 49

3.5 Spectral Convolution 57

3.6 Noise 58

3.6.1 Simulating noise 61

3.7 Summary 66

Chapter 4: A Comparison of Mapping Arctic Lithologies using Simulated EnMap and Airborne Hyperspectral Imagery 68

4. 2 Introduction 68

4.3 Study area 70

4.4 Methods 72

4.4.1 Data and preprocessing 72

4.4.1 EnMap Simulation 73

4.4.2 Data Processing - Image Preparation 76

Masking 76

Band Removal 78

Noise Reduction (Inverse MNF) 78

Band Ratios 80

Reference data preparation 82

4.4.3 Supervised Classification of Lithological Units 85

Matched filtering classification for Lithologic map units 85

Random Forest classification for Lithologic map units 86

4.5 Results 88

4.5.1 Training Area Spectral Signatures 88

4.5.2 Evaluation of Training Area Separability 91

4.5.3 Matched Filtering Classification 96

Spatial Filtering of Matched Filtering Abundance Images 99

4.5.4 Random Forest Classification 102

4.5.5 Differences in modelled Point Spread Function (PSF) 110

4.6 Discussion 112

4.7 Conclusion 121

Chapter 5: Conclusion and Recommendations 123

5.1 Suggestions for further research: 128

References 131

Appendix A: Acquisition Table of Flight Dates and Times 155

Appendix B: CRAN R Scripts for Sensor Simulation 156

Appendix C: CRAN R Script for Thresholding Matched Filtered Score Images and Conversion to a Multiclass Map 189

Appendix D: Sherlock and Carpenter's Map (2003): Bedrock Geology of the Wolverine-Doris Corridor, Hope Bay Volcanic Belt, Nunavut (GSC Open File 1553) 193

List of Figures

Figure 1.1: Similar JPL spectral library signatures for alunite (SO-4A) (in red), kaolinite (PS-1A) (in green), and muscovite (PS-16A)(in blue) as they are captured using hyperspectral imaging as compared with the multispectral satellites ASTER and Landsat 7..... 5

Figure 2.1: The geometry of the sun and sensor view may vary greatly by both their azimuth (θ) and altitude (ϕ) as shown in this diagram. (Source: Kerekes and Landgrebe, 1989)..... 17

Figure 2.2: Specular and diffuse reflection (Source: Zheng et al., 2012). 19

Figure 2.3: Adjacency effect shown in photon reflectance paths 3 & 4 (source: Richter et al., 2006). 22

Figure 2.4: Four possible interactions with atmospheric molecules and aerosols (source: Aria, 2013). 24

Figure 2.5: Continuum removed spectral signatures for the mineral kaolinite with decreasing spectral resolutions. Note the loss of the doublet absorption feature at 2180 nm (source: Kaufmann et al., 2011). 33

Figure 2.6: Keystone and smile (Source: Yokoya et al., 2010). 39

Figure 3.1: General processing workflow..... 47

Figure 3.2: A 2-dimensional plot of the sinc function for a PSF with a FWHM equal to 10 pixels of the input data. Observe the infinite outward lobes. Grey crosses mark the GIFOV and FWHM. 51

Figure 3.3: A 3-dimensional plot of the sinc function demonstrating the negative (Y-axis) influence of the signal with the outward lobes. 51

Figure 3.4: A 2-dimensional plot of the Gaussian function for a PSF with a FWHM equal to 10 pixels of the input data. The location of the FWHM (determining the spatial resolution of the simulated sensor) is shown in light grey crosses. The ideal sinc PSF is shown in the light grey dashed line for comparison. 53

Figure 3.5: A 3-dimensional plot of the Gaussian function. 53

Figure 3.6: A 2-dimensional plot of the hamming window function shown in green and the result of multiplying this window by a sinc window function in blue. Note how the sinc-hamming window function terminates at 0 making it more practical to use than a sinc function terminates at 0 making it more practical to use than a sinc function. 55

Figure 3.7: A 3-dimensional plot of the sinc-hamming function. 55

Figure 3.8: The Spectral Response Function of an individual 10 nm wide band plotted in 2-dimensions and shown in black. Grey lines represent the SRF of neighbouring bands. Circles represent weighted values used in the simulation at 1 nm intervals. . 58

Figure 3.9: A 3-dimensional plot of the simulated noise. The upper surface is the signal dependent noise while the slightly obscured lower surface is the signal independent noise. 65

Figure 4.1: Location map showing the Hope Bay region in relationship to the rest of Canada (upper left) and the area covered by the ProSpecTIR–SPECIM hyperspectral survey used for EnMap simulation. 72

Figure 4.2: General processing workflow used for simulation of EnMap simulation through to lithological unit classification. 74

Figure 4.3: Three modelled PSFs with the ideal sync function (in grey) for reference. The Gaussian is shown in red, the sinc-hamming in blue, and the pixel aggregate in orange. Although not used as a modelled PSF, the hamming window filter is shown in green to demonstrate how it appears before multiplying it with the sinc window function. 76

Figure 4.4: MNF Eigenvalue plots for the Wolverine-Doris Corridor AISA ProSpecTIR-SPECIM (top) and EnMap (bottom) imagery. For both sensors the top 50 Eigen values (components) were used in the inverse MNF transform. 80

Figure 4.5: Averaged spectral signatures obtained from the training areas for each lithological class extracted from the ProSpecTIR–SPECIM (top) and the simulated EnMap (bottom). Atmospheric absorption bands have been removed..... 90

Figure 4.6: Box and whisker plots of Transformed Divergence (TD) values for all lithological classes using all band ratios and indices for (a) simulated EnMap (Gaussian PSF) on the left and (b) ProSpecTIR–SPECIM data on the right. Plotted are the mean value as the thick central line, min and max values as whiskers, and quartiles as the box. 92

Figure 4.7: Standard deviation values found in each class training dataset for the band ratios and indices for both the EnMap (Gaussian PSF) and ProSpecTIR–SPECIM data (boxes). Also shown are the class accuracies for the RF classification using this data (lines). 94

Figure 4.8: Standard deviation values found in each class training dataset for the inverse MNF imagery for both the EnMap (Gaussian PSF) and ProSpecTIR–SPECIM data (boxes). Also shown are the class accuracies for the MF classification using this data (lines). 95

Figure 4.9: Comparison of the generalized published geology map (left) to that of the spatially filtered MF classification results for the simulated EnMap (Sinc-Hamming PSF) scene (center) and ProSpecTIR–SPECIM data (right). Map scale is roughly 1:100,000. 104

Figure 4.10: Comparison of the generalized published geology map (left) to that of the RF classification results for the simulated EnMap (Sinc-Hamming PSF) scene (center) and ProSpecTIR–SPECIM data (right). Map scale is roughly 1:100,000. 105

Figure 4.11: Graph of the changing overall accuracies and kappa coefficients with differing filtered window size for the classified maps based on Matched Filtering scores. At top are the values for the ProSpecTIR–SPECIM data and at bottom are the simulated EnMap (u sing a Sinc-Hamming PSF) results. 106

Figure 4.12: The error rate with the number of trees grown in the RF classification for the ProSpecTIR – SPECIM derived ratios and indices. 109

Figure 4.13: The error rate with the number of trees grown in the RF classification for the EnMap derived ratios and indices. 109

Figure 4.14: True colour images showing the visual differences between the EO-1 Hyperion sensor imagery (July 2013), EnMap simulated from airborne data (July, 2014) using different modelled PSFs. Map scale is roughly 1:35,000. 112

Figure 4.15: Visual loss and simplification of the classification when spatially filtered MF results are applied through increasing kernel sized spatial filters. EnMap classifcation results are also shown for comparison. Map scale is roughly 1:40,000. 113

Figure 4.16: Semivariograms for five wavelength positions (top) and band ratio and indices (bottom) from the 3m ProSpecTIR - SPECIM image. 115

Figure 4.17: Class mean spectral signatures as they appear in the multispectral sensors ASTER and Landsat 8. 118

Figure 4.18: Variable importance plots showing the mean decrease in accuracy (top) and Gini scores (right) from the RF classification of the ProSPecTIR – SPECIM (left) and the EnMap imagery (right). 119

Figure 4.19: RGB composite of the three highest variable importance bands (R: Ferric Iron, G: Kaolinite, and B: Gossan) from the RF classification (EnMap) before thresholding probability values. Areas shown in black represent masked vegatation and water. 120

List of Tables

Table 1.1: Current planned spaceborne hyperspectral imagers and their key performance parameters. 8

Table 4.1: List of common ASTER and Landsat band ratios for targeted mineralization spectral features. Included are the wavelengths for which they were created for and the corresponding bands used in this study. 82

Table 4.2: Lithological unit classes and matching training and verification sizes (number of polygons/number of pixels) for both datasets. 84

Table 4.3: Confusion matrix for EnMap (Aggregation PSF) MF classification results. 100

Table 4.4: Confusion matrix for EnMap (Gaussian PSF) MF classification results. 100

Table 4.5: Confusion matrix for EnMap (Sinc-Hamming PSF) MF classification results. 101

Table 4.6: Confusion matrix for ProSpecTIR-SPECIM MF classification results..... 101

Table 4.7: Confusion matrix for ProSpecTIR-SPECIM MF classification results with 17x17 spatial filtering applied. 101

Table 4.8: Confusion matrix for EnMap (Aggregation PSF) RF classification results.. 107

Table 4.9: Confusion matrix for EnMap (Gaussian PSF) RF classification results. 107

Table 4.10: Confusion matrix for EnMap (Sinc-Hamming PSF) RF classification results. 107

Table 4.11: Confusion matrix for ProSpecTIR-SPECIM RF classification results..... 108

Table 4.12: MF Classification accuracies as a function of the modelled PSF..... 111

Table 4.13: RF Classification accuracies as a function of the modelled PSF..... 111

List of Abbreviations

Au	Gold
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
CCD	Charge-coupled devices
CHRIS	Compact High Resolution Imaging Spectrometer
CMOS	Complementary metal-oxide semiconductors
CRAN	Comprehensive R archive network
dB	Decibels
DN	Digital number
DPI	Dots per inch
DR	Dynamic range
EO-1	Earth Observer-1
EM	Electromagnetic
EnMap	Environmental Mapping and Analysis Program
FOV	Field-of-view
FWHM	Full width at half maximum
DLR	German Aerospace Center
GCP	Ground control points
GIFOV	Ground-projected instantaneous field-of-view
GSC	Geological Survey of Canada
GSD	Ground Sampling Distance
HgCdTe	Mercury-cadmium-telluride
HSI	Hyperspectral imager
IFOV	Instantaneous-field-of-view
JPL	Jet Propulsion Lab
K	Kappa coefficient
MERIS	MEdium Resolution Imaging Spectrometer
MF	Matched filtering

MNF	Minimum noise fraction transformation
MODTRAN	MODerate resolution TRANsmission model
NASA	National Aeronautics and Space Agency
NDVI	Normalized difference vegetation index
nm	nanometers
NWT	Northwest Territories
PC	Principal component
PRISMA	PRecursore IperSpettrale della Missione Applicativa
PSF	Point spread function
QE	Quantum efficiency
RF	Random forest
RGB	Red green blue
ROI	Region of interest
RMS	Root mean square
RPM	Remote predictive mapping
S2A	Sentinel-2A
SAR	Synthetic aperture radar
SINED	Strategic Investment in Northern Economic Development
SiO	Silicone monoxide
SNR	Signal-to-noise ratio
SPOT	Satellite Pour l'Observation de la Terra
SRF	Spectral response function
TIR	Thermal infrared
TOC	Topographic correction
TD	Transformed divergence
μm	micrometers
VMS	Volcanic massive sulphides

Chapter 1: Introduction

1.1 Background

Conventional geological field mapping programs in the Canadian Arctic are challenged by high operational costs. The vast and remote locations of these sites mean that getting to the field study areas, setting up field camps, and utilizing helicopters to traverse large study regions makes for higher expenses than other parts of the world as it lacks the existing transportation and infrastructure to support such activities. Despite the economic potential of natural resources in the region, along with the opening of sea ice due to climate change that is ensuring new navigable transportation routes, there are still regions previously unmapped beyond coarse scales. This scale and quality of mapping is disproportionate to that of the rest of Canada. In response, territorial and national government agencies and mineral exploration companies are currently striving to obtain better geoscience knowledge to support economic and infrastructure development.

Remote sensing data and thier associated analytical methods are useful tools for assisting field geology campaigns as they have the potential to assist in producing geological maps more quickly and efficiently. Although predictive in nature, remote sensing data have been well established for providing first order geological information of a given study area that can help establish the locations for field follow-up and help avoid areas that are either geologically homogenous or covered by either over-burden or vegetation and thus don’t require “boots-on-the-ground” investigation. This information thereby reduces cost, crew sizes, and time requirements (Harris et al., 2006; Harris, 2007) but is not a

replacement for field mapping as both are required to produce geological maps of Canada's North.

Remote sensing imagery obtained by either aircraft or satellites leads to the potential of objective, measurable, and controlled mapping that can augment pre and post-fieldwork geological mapping. When geologists are unable to cover entire study areas the remaining parts can be augmented with results obtained from remotely sensed imagery.

Of the many remote sensing technologies that are available to be utilized (magnetometers, gamma-ray spectrometers, synthetic aperture radars (SAR), density gravimeters, and thermal radiometers) optical and infrared spectral reflectance sensors have shown to be particularly successful for lithological mapping. For decades studies effectively utilized multispectral optical sensors in dry, mostly unvegetated environments with distinct geological regions using ASTER and Landsat data (Crippen & Blom, 2001; Rowan et al., 2003; Rajesh, 2004; Zhang and Pazner, 2007; Gomez et al., 2008; Rockwell & Hofstra, 2008). Studies have demonstrated their use in the less ideal arctic environment. These arctic studies identified that spectrally mixed lithologies, snow cover, pervasive lichen and overburden, and low sun azimuths which produce topographic shadows and reduce measureable reflectance pose challenges to accurate obtain geoscience information in this environment (Schetselaar and de Kemp, 2000; Budkewitsch et al. 2000; Staenz et al., 2000, 2001; Lorenz, 2004; Wickert and Budkewitsch, 2004; Harris, 2008; Rivard et al., 2010; Behnia et al., 2012).

These geological mapping studies have largely relied on multispectral satellite sensors that measure broad bands in the Very Near Infrared (VNIR), Short Wave Infrared

(SWIR), and Thermal Infrared (TIR) wavelengths. The rationale is that these wavelength ranges contain diagnostic spectral features that differentiate rocks based on their various mineral reflectance and emissivity properties. Each of these sections (VNIR, SWIR, and TIR) is influenced by differing rock properties. The shape of the spectral signatures within the VNIR wavelengths are mostly affected by the interaction of energy levels of electrons (electronic transitions) and are strongly effected by crystal field effects on translational energy levels with ions within elements (Hunt, 1977). Iron oxide minerals substantially influence the shape of spectra in the VNIR wavelength range (Hunt and Ashley, 1979). The spectral signatures within the SWIR spectrum are largely shaped by vibrational processes of atoms within minerals. The reflectance properties of a number of minerals have narrow absorptions features that permit their identification particularly in the 2.0 to 2.4 μm region of the EM spectrum (Goetz et al., 1985). This helps with the identification of specific absorption features associated with carbonates, sulphates, micas, and clay based minerals (Clark et al., 1990). Some of the spectral signatures of these minerals have very narrow distinctive absorption features that require fine spectral measurement to delineate. To demonstrate, the VNIR and SWIR spectral signatures of the clay minerals: alunite, kaolinite, and muscovite measured from lab imaging spectrometers are well known to be nearly identical except for subtle differences between 1.4 and 1.5 as well as 2.12 and 2.35 μm (Clark et al., 1999). Figure 1.1 displays NASA JPL spectral libraries signatures for these minerals in the 400-2500 wavelength ranges: alunite in red, kaolinite in green, and muscovite in blue.

Lastly, the spectra of rocks in the TIR wavelengths are related to the fundamental vibrations of silicon monoxide (SiO) content and are principally useful for differentiating rocks with high silicate and carbonate content (Lyon, 1965).

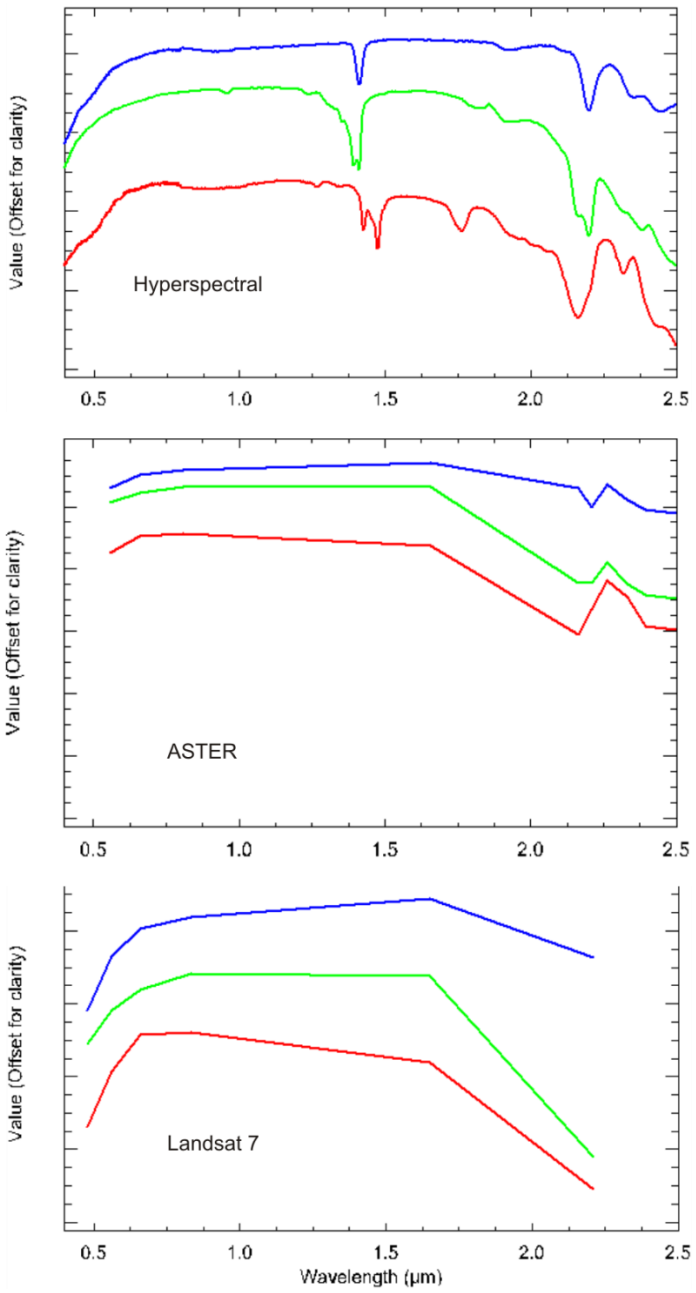

Figure 1.1: Similar JPL spectral library signatures for alunite (SO-4A) (in red), kaolinite (PS-1A) (in green), and muscovite (PS-16A)(in blue) as they are captured using hyperspectral imaging as compared with the multispectral satellites ASTER and Landsat 7.

Despite the effective use of multispectral sensors the most substantial limitation found in these studies is that any of these sensors that measure the SWIR and TIR wavelengths are

technologically limited to a so called “moderate” spatial resolution. This moderate resolution, typically around 15-90 m, frequently becomes problematic as the pixels represent a mixture of many surface materials and if the lithological classes of interest share similar, highly variable spectral signatures, contain narrow diagnostic spectral features, and/or are dominated by non-geological materials (such as vegetation or water) attempting to classify the imagery likely results in poor prediction accuracies. The lower spectral coverage and resolution of these multispectral sensors also causes materials with similar spectral signatures to become ambiguous to delineate. Figure 1.1 demonstrates this situation by showing how important characteristic spectral absorption features of alunite, kaolinite, and muscovite are lost when resampled to resolutions of Landsat 7 and ASTER.

Concurrent with the advancement of multispectral sensors has been the recent development of hyperspectral imagers (HSI). This technology provides more detailed image analysis which overcomes the issue of poor classification accuracies for regions with sufficiently heterogeneous landscapes (Staenz, 1992). Unlike multispectral imaging systems which measure the Earth’s surface reflectance in a low number of separated and wide wavelength bands, HSI systems measure the EM spectrum at a series of narrow and contiguous spectral wavelength bands usually in the hundreds to produce a profile of the spectral reflectance of measured surfaces. The justification for this augmentation in imaging technology is that the increases in spectral coverage and resolution helps distinguish classes within mixed pixels: those that cover ground consisting of multiple surface types or map classes. The acquired narrow and contiguous spectrum for pixels provides a spectral signature with sets of diagnostic features that individually are

associated with targeted map unit classes or materials. The result is an ability to classify, detect, or unmix pixels in heterogeneous landscapes. An added benefit of the contiguous spectral coverage is the ability to obtain spectral absorption/reflectance features otherwise not measured/missed using a multispectral sensor. Where pixels are difficult to classify in multispectral imaging, HSI imagery is able to better classify pixels in heterogeneous pixels and thereby the classification process doesn't necessitate increased spatial resolution.

Hyperion, onboard the technological testbed satellite: Earth Observer-1 (EO-1) was the only hyperspectral satellite spectrometer capable of measuring the VNIR and SWIR wavelengths. At the time of its launch, Hyperion was designed as an evaluation of the idea of a spaceborne hyperspectral imaging spectrometer (Pearlman et al., 2001) and as a result had limited practical abilities. Despite its reported poor image fidelity, Hyperion had shown to improve classification performances for lithological mapping when compared to similar spatial resolution multispectral sensors (Zhang and Pazner, 2007; Leverington, 2010; Dadon et al., 2011). Nevertheless its relatively low Signal-to-Noise Ratio (SNR) and narrow swath (7.5 km) had severely limited its capability as a workhorse sensor for geoscience mapping. International interest in the spaceborne hyperspectral concept has continued as a result of Hyperion and is demonstrated in the development or planning of new spaceborne sensors (see table 1.1) with significantly improved SNR and ground swath coverage that will potentially result in widespread use of such a sensor (Staenz & Held, 2012).

Table 1.1: Current planned spaceborne hyperspectral imagers and their key performance parameters.

Sensor	Hyperion	PRISMA	HISUI	EnMap	HypXIM-P	SHALOM	HyspIRI
Country	USA	Italy	Japan	Germany	France	Italy/Israel	USA
Spatial resolution (m)	30	30	30	30	16	8 (VNIR) 12 (SWIR)	30
Swath at nadir (km)	7.5	30	30	30	<8	>10	30
Wavelength coverage (nm)	357-2576	400-2500	400-2500	420-2450	400-2500	400-2400	380-2500
Number of bands	220	237	185	242	210	>200	>200
Spectral resolution (nm)	10	10	10 (VNIR) 12.5 (SWIR)	6.5 (VNIR) 10 (SWIR)	10	10	10
SNR	150:1 (VNIR) 50:1 (SWIR)	>200:1 (VNIR) >100:1 (SWIR)	>450:1 (VNIR) >300 (SWIR)	500:1 (VNIR) 150:1 (SWIR)	250:1 (VNIR) >100:1 (SWIR)	>200:1 (VNIR) >100(SWIR)	~400:1
Detector	CCD / HgCdTe	Unspecified	CMOS/HgCdTe	CMOS/HgCdTe	Unspecified	Unspecified	Unspecified
Launch Date	2000	2018	2018	2019	>2020	2021	>2022

Sources: Pearlman et al., 2001, Buckingham and Staenz, 2008; Briottet et al., 2011; Staenz and Held, 2012; van de Meer et al., 2012; Matsunaga et al., 2013; Lopinto and Ananasso, 2013; Guanter et al., 2015; Lee et al., 2015; Feingersh and Dor, 2015

The spaceborne hyperspectral imaging (HSI) spectrometer sensor that is currently the furthest most developed is the German Aerospace Center (DLR) satellite mission: Environmental Mapping and Analysis Program (or EnMap for short). After its approaching launch in mid-2019 EnMap's image availability is anticipated to take an open data policy (Guanter et al., 2015). This accessibility of free VNIR and SWIR imagery will no doubt result in its wide use in geosciences and elsewhere.

Significant interest exists in determining the geological mapping potential of EnMap imagery, the limitations of coarser spectral/spatial resolutions and increased noise characteristics for comparable future hyperspectral satellites (Staenz et al., 2005; White et al., 2010; Rogge et al., 2014). Selection of the most optimal sensor and corresponding methods of data analysis helps geologists and remote sensing practitioners achieve accurate results.

However conducting an assessment of spaceborne VNIR and SWIR hyperspectral data is challenging given that there are no data to use for the assessment. The only satellite that

measured this spectral coverage, Hyperion, is now decommissioned, and its archived data are characterized by inadequate SNR. This thesis uses the Comprehensive R Archive Network (CRAN) for the R programming language to develop a modeling script to synthesize an EnMAP image scene. The script was written to convolve oversampled spectral and spatial resolution airborne hyperspectral imagery to that of the expected EnMap sensor resolutions. The sensor's Point Spread Function (PSF), Spectral Response Function (SRF), and signal dependent and independent noise levels were modelled. Although many other performance factors affect the capability to derive geological mapping from hyperspectral imagery including: spatial and spectral aberrations, and increased optical depth, the spectral and spatial resolution, spectral coverage, and SNR are identified to be the key factors in influencing their capability to successfully predict geological map units (Cushnie et al., 1987; Clark et al., 1990; Kruse, 2000; Swayze et al., 2003; Harris & Wickert, 2008).

This research evaluated the applicability of EnMap to map bedrock lithologies for a characteristic and well mapped area of the Canadian Arctic (Hope Bay Greenstone Belt, Northwest Territories). The region consists of six main lithologies, consisting largely of mafic and felsic volcanics along with sedimentary and volcaniclastic rocks. The results along with those derived from the high spectral and spatial resolution airborne data used to produce the simulated EnMap imagery are compared to that of a detailed 1:25,000 geological map.

Similar previous work (White et al., 2010; Rogge et al., 2014) was done to compare and evaluate EnMap and airborne hyperspectral (AISA) imagery for their ability to produce

detailed predictive lithologic and mineral abundance maps. Both studies were conducted in Canada's North using a technique called image endmember extraction, a process by which one pixel thought to be a pure representation for each targeted material is found through analysis of the imagery. The Rogge el al. (2014) study showed that both the AISA and EnMap imagery could successfully discriminate map units however EnMap was less able to relate predicted units to those found in regional geological maps. This was thought to be caused by its lower spatial and spectral resolutions. This conclusion could also be the result of the thoroughness and density of the geological mapping used for comparison as Arctic maps are often published with less detail.

This thesis is an extension of these earlier studies, however it takes a different approach to the classification: it uses multiple homogenous outcrop areas based on existing geological mapping within the imagery as training, rather than image endmember extraction and uses statistical confusion matrices to evaluate the accuracy of the predicted mapping. Justification for this different approach is based on the observation made in previous studies that accurate endmember extraction is challenging for lithological units particularly at coarser resolutions (Staenz, et al., 2001). Even within high resolution imagery most pixels are a mixture of materials and obtaining a so called *pure* endmember is ambiguous. At coarser spatial resolutions, such as the 30 m resolution of EnMap, the composition of each pixel is potentially a larger mixture of spectral signatures from several materials including overburden and water. Additionally, determining the *pure* endmember may also be challenging given that lithological units contain a high degree of spectral variability caused by mixed mineralogy, rock surface weathering, grain size differences, and preferential lichen and/or vegetation cover. Harris et al. (2006) through

comparing supervised classification results using image derived endmember extraction versus training areas for representative class spectra found that using training areas provided more optimal results albeit requires prior knowledge of the geology of the site either through past mapping or new fieldwork.

1.2 Research Goals and Objectives

The main goal of this research was to ascertain whether lithological units could be differentiated and classified to produce a predictive geological map using airborne and the next generation of spaceborne remotely sensed hyperspectral imagery. Another goal was to develop and determine an optimal classification approach. Specifically the research questions designed to address these goals are as follows:

1. Can VNIR/SWIR hyperspectral imagery be effective in classifying spectrally similar lithological units (mafic volcanics, felsic volcanics, sedimentary, and volcaniclastic rocks) in the Hope Bay Greenstone Belt of the Northwest Territories?
2. What effect does lower SNR and spatial and spectral resolution associated with simulated EnMap imagery have on accuracy compared to that of airborne hyperspectral imagery?
3. Can the choice between using established geological band ratios alongside the Random Forest (RF) classifier algorithm and thresholding Matched Filtering scores improve the accuracy of Remote Predictive Maps (RPM)?

4. Finally, to better understand the impact of simulation procedures, what effect does the choice in kernel filters for modelling a coarser sensor's Point Spread Function (PSF) have on lithological map unit classification accuracy?

To answer these questions the following detailed objectives were defined:

1. To simulate a partial EnMap scene. This was achieved by writing a CRAN R script that models EnMap's spatial PSF, Spectral Response Function (SRF), and signal dependent and independent noise levels to produce a simulated scene.
2. In an effort to determine the optimal sensor, evaluate and compare the accuracy of predictive maps derived from two different supervised classification approaches for both the airborne and simulated EnMap imagery.
3. Ascertain the impact, both statistically and visually, of different spatial convolving/resampling approaches for PSF modelling.

1.2 Thesis Structure

This thesis consists of five chapters including this introductory chapter that provides an overview of the background and states the research goals and objectives.

Chapter 2 is titled "Factors Influencing the Accuracy of Predicted Lithological Units through the use of Hyperspectral Imagery". It serves as a literature review on the limitations of remote predictive mapping using optical and shortwave infrared hyperspectral technology.

Chapter 3 titled “Simulation of EnMap data through upscaling of airborne HSI data” documents the theoretical framework and rational for the simulation of a hyperspectral satellite image and reviews how the simulation of the EnMap scene used in the thesis was derived.

Chapter 4 documents how the EnMap simulated scene and ProSpecTIR–SPECIM (a proprietary modified AISA sensor) airborne data were used to map lithological units and ascertain the optimal sensor and approach to working with these data. Training sites of identifiable outcrops were determined from the high resolution ProSpecTIR–SPECIM imagery. Their class definitions for each site were then derived from an existing detailed geological bedrock map. These sites were then used to conduct a supervised classification of the imagery using two very disparate approaches: 1) by thresholding Matched Filtering scores derived from forward and successive inverse minimum noise fraction (MNF) images and 2) by using the Random Forest algorithm on several known geological band indices. Confusion matrices using verification Regions of Interests (ROIs) also derived from the detailed geological map were used to assess the accuracy of the predicted maps using the following statistical measures: 1) overall accuracy, 2) commission and omission error, 3) producer accuracy, and 4) user accuracy for each class and 5) Kappa coefficient (k).

Chapter 5 provides a summary of the thesis. This chapter also discusses the study’s implications for Arctic geological mapping, and includes a number of recommendations for future work.

An appendix at the end of the thesis contains CRAN R scripts used for simulating the EnMap hyperspectral satellite scene and for the conversion of matched filtering scores to a hard classification. Additionally, the appendices contain supplementary information about the dates and times of the acquisition of the airborne hyperspectral imagery employed in this study.

Chapter 2: Factors Influencing the Accuracy of Predicted Lithological Units Through the use of Hyperspectral Imagery

2.1 Introduction

Classification of optical and infrared spectral reflectance remote sensing imagery involves converting reflectance measurements made by a sensor into cartographic thematic map units (Jensen, 2007). This conversion from measurements into cognitive representations of the Earth's surface is inherently flawed. Despite recent advances in the development of sensor technology, an imaging system is still constrained by several parameters that, as a result, do not accurately capture the real world. Additionally, the natural variations of the environment cause problems for generalizations that are required for representation of abstract thematic map units. The accuracy of the classification of remotely sensed data are hence greatly dependent on an overwhelming number of factors.

With the aim of obtaining the highest classification performance for mapping, remote sensing scientists and signal processing engineers who design these systems have built a large body of research identifying and documenting the impact of different factors that influence accuracy. This work has identified that classification accuracy is a function of the sensor capabilities, scene geometry, the nature of the atmosphere, landscape, and classification algorithm, and training/verification data employed.

Before comparing relative hyperspectral sensor imagery capabilities for lithological mapping later in this thesis, it is prudent to have an in-depth review and understanding of each of these factors. The goal of this chapter therefore is to identify the most significant influences, briefly explain them, and discuss the degree of their relevance derived

through past empirical studies. This chapter subdivides the factors into three subgroups: scene geometry, sensor specific influences, and processing considerations.

2.2 Scene Specific Considerations

2.2.1 Scene Geometry

Hyperspectral imagers are optical passive sensors, and as such need energy in the form of reflected light photons. The imaging system's array receives these photons and converts them to a digital signal. During the path from incoming solar energy into the Earth's atmosphere and to the measurement by the spectrometer array, the photons are subjected to a number of highly variable geometric processes. Two such geometries are the sun and the sensor altitude (zenith) and azimuth (see figure 2.1). Light energy is also affected by the orientation of the objects it reflects off, varied by both the angles of slope and azimuth of these surfaces. These highly variable geometric differences not only influence the overall amount of available energy to a sensor array but also can create localized variable light reflectance. The effect of these varying geometries can influence the spectral signatures of materials.

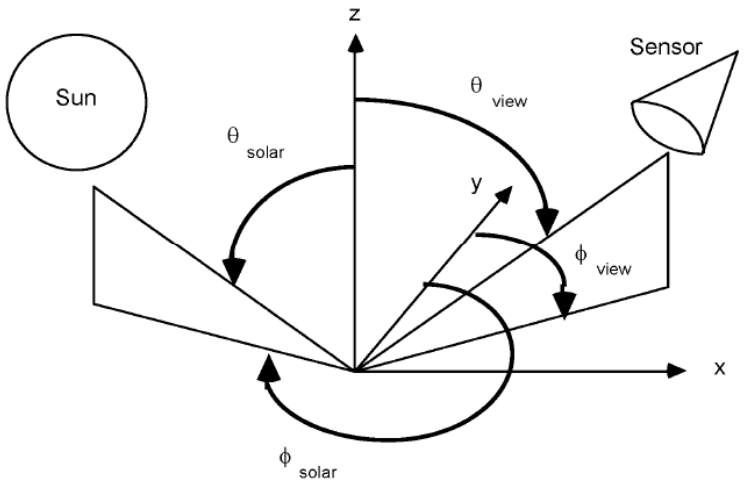

Figure 2.1: The geometry of the sun and sensor view may vary greatly by both their azimuth (θ) and altitude (ϕ) as shown in this diagram. (Source: Kerekes and Landgrebe, 1989)

To demonstrate, if during image acquisition there is a low solar zenith there may be inadequate illumination to effectively differentiate features. Jensen (2007) argues that the sun zenith needs to be greater than 30° to provide adequate surface reflectance for mapping. Harris et al. (2014) noted this detrimental effect and how problematic it is in arctic environments where opportunities for image acquisition with moderately high sun azimuths are rare. They found that when comparing ASTER and Landsat multispectral imagery, despite ASTER's relatively greater spectral coverage and resolution, Landsat performed better. The overall lithological classification accuracy had an increased difference of 19.1% when classifying the raw bands for Landsat imagery. This was attributed to a low solar zenith (20°) during the ASTER acquisition time. Neville et al. (2003) also noted that sun illumination differences between two sensors imaging the same area lead to amplitude differences in absorption features of spectra. The decreased albedo from targeted minerals led to disparities in the extracted spectra representing pure

concentrations of materials. Furthermore, the reduction in the depth and height of spectral features also reduces the ability to determine non-mixed pixels (Harris, 2005). Determining non-mixed pixels is essential when attempting to discover well represented endmembers for unmixing other less homogenous pixels. A low sun azimuth is also responsible for decreased image contrast that reduces the signal to noise ratio (SNR). Inadequate SNR is the most reported factor influencing the lower accuracies for the hyperspectral satellite Hyperion (Kruse et al., 2000, 2003) especially when the sun zenith is lower than nominally 35 degrees (Kruse et al., 2002; Kuosmanen et al., 2005; Dadon et al, 2011).

Low sun azimuths further increase the length of shadows from scene objects such as vegetation and topographic features. Such shadows are either acknowledged and left untreated (Harris et al., 2005; Behnia et al., 2012) or masked and thereby lose the potential to map these areas (Harris et al., 2014).

Sensor view geometry, not to be confused with solar geometry previously mentioned, can be responsible for overall increases/decreases in image spectral contrast. Spectral signatures and thereby the absorption depths can vary depending on the sensor viewing angles. Lorenz (2004) noted that there were pronounced contrast differences between forward looking and backward looking imagery (15° from nadir) taken from the Corona satellite despite the images being acquired seconds apart and with the same solar illumination conditions.

Sun and sensor view geometries are also affected by two important reflectance properties of scene materials: specular and diffuse reflection (see figure 2.2). Most landscape

materials exhibit varying degrees of both (Zheng et al., 2012). Specular reflection refers to the law of reflection that states that when light rays strike a surface they are reflected outwards at the same angle with reference to the surface's normal plane (Taylor, 2000). Diffuse reflection, on the other hand, refers to light energy when reflected off an object does so equally in all directions in accordance with Lambert's cosine law (Taylor, 2000). Smooth surfaces increase specular reflection whereas rough surface exhibit diffuse reflection properties (Taylor, 2000; Zhao et al., 2010). Rough surfaces obey the law of reflectance associated with specular reflection, but because rough surfaces contain microscopic scale variations in the surface planes they give the property of reflected light in all directions at the macroscopic level (Taylor, 2000).

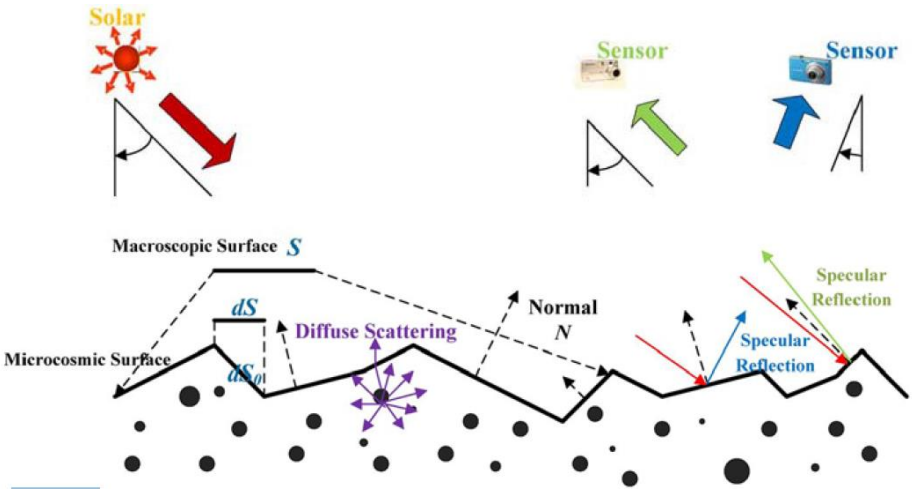

Figure 2.2: Specular and diffuse reflection (Source: Zheng et al., 2012).

Diffuse and specular reflection properties influence how a scene is illuminated and thereby how detectable pixel spectra are represented. For example, a surface material type with bright diffuse reflection properties, such as bright lightly coloured rocks and another with high specular reflection properties, such as calm water, will illuminate the

backsides of nearby sun shaded slopes in separate manners due to the reflectance properties of these two materials. Specular reflected rays from the water will have greater focused directional influence determined by the solar zenith and azimuth. The bedrock will provide more uniform illumination of the surrounding area (diffuse inter-reflection) and is the more common influence on surrounding terrain affecting geological studies (J. Harris, personal communication, July 17, 2017). The albedo intensity properties of surface materials will also greatly influence the strength of these reflectance effects.

Airborne and satellite optical systems are designed to have limited or fixed pointing directions at or close to nadir, and thus intense specular reflectance becomes significant when the sun zenith is high and closely matching the sensor viewing angle. For this reason, Jensen (2007) suggests the solar zenith be less than 52° (and greater than 30° to ensure adequate illumination) during image acquisition times. This large difference between sensor and sun geometry (38°) is justifiable since intense specular surfaces can contain a large amount of variable surface planes. For example, a wind disturbed water surfaces will create intense bright spots in the imagery (termed hot spots or sun glint) even with a significant difference between sensor and solar zeniths.

Topographic variability also increases the probability of specular reflectance. Sloped terrain will change the surface geometries such that specular reflectance can be directed towards the sensor. Surface slope angle additionally influences the amount of multiple reflections on the ground. When two neighbouring slopes face one another there is a known effect of increased photon scattering between the slopes (Arai, 2013). This decreases the amount of available radiance for the instrument to measure as the average

number of photons is decreased in relationship to decreased angles between slopes (Arai, 2013).

Yet another factor influencing spectral variability of targeted materials, caused by scene geometry, is the *adjacency effect*. This refers to an occurrence where the spectral content of a single pixel can be contaminated with spectra of neighbouring pixels. The adjacency effect is caused when the reflectance of photons from neighbouring pixels are scattered by atmospheric water vapour and aerosols but into the sensor's instantaneous-field-of-view (IFOV) of the neighbouring pixel (see path radiances 3 and 4 in figure 2.3) (Otterman and Fraser, 1979; Richter et al., 2006). High albedo materials increase the effect especially when the neighbouring pixels have low albedo intensity, or in other words with increased contrast between neighbouring pixels (Otterman and Fraser, 1979; Richter et al., 2006). The influence of the adjacency effect is not uniform across the spectrum; it has been observed to be stronger with higher wavelengths, dominantly in the 400-1000 nm region (Richter et al., 2006). Atmospheric correction algorithms infrequently attempt to reduce the adjacency effect (Burazerović et al., 2013).

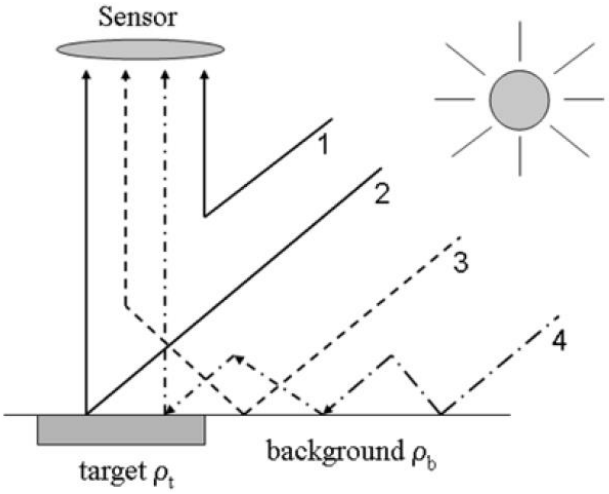

Figure 2.3: Adjacency effect shown in photon reflectance paths 3 & 4 (source: Richter et al., 2006).

To summarize, variations in several scene properties: solar, sensor, and surface geometries, specular/diffuse properties from neighbouring pixels, albedo, and wavelength dependencies all lead to spatial variability and unpredictability in the spectral signatures of materials. Classification algorithms rely on fixed spectral signatures and thus their prediction performance is reduced with variation. Topographic correction (TOC) techniques are used to minimize the effects of varied geometries (Achard and Lenot 2009; Richter et al., 2009). However TOC is computationally demanding, is not always applied, and depending on the technique applied out of a need for modelling simplicity assume that the scene surfaces are a true Lambertian (Sola et al., 2014). Modeling to reduce scene geometry illumination variations is additionally complicated given that the variability of surface reflectance between different and within materials is highly complex and poorly understood (Zhang et al. 2010).

2.2.2 Atmospheric Considerations

The reflectance values of imagery pixels are variably influenced by interactions of light photos with the atmosphere during their path from solar irradiance, surface reflectance, and to measurement. Photons, in the course of their transmittance, are subjected to one of four possible interactions with atmospheric molecules and aerosols (see figure 2.4). Arai (2013) describes these as photons being diffusely scattered by atmospheric molecules and aerosols and then being: 1) reflected out to the atmosphere, 2) absorbed into the atmosphere, 3) reflected off the ground (and possibly being scattered again) and out of the atmosphere, or 4) absorbed into the Earth's surface. The amount of light transmittance through the atmosphere (termed optical depth) is influenced by meteorological conditions and the aerosol particles in the atmosphere. All these photon passes are unfavourable to classification efforts as they decrease the amount of measurable radiance and increase the noise of spectra. The passes where photons are scattered and interact with the Earth's surface or scattered after interacting with the Earth (passes 3 and 4 in figure 2.4) lead to the adjacency effect discussed earlier.

Photons scatter in the atmosphere due to two possible scattering laws: Rayleigh and Mie according to properties of the molecules with which they interact. Scattering by particles smaller than the wavelength leads to Rayleigh scattering whereas scattering by particles larger than the wavelength leads to Mie scattering (Jensen, 2007). Aerosols typically scatter or absorb light according to the Mie law whereas atmospheric molecules scatter photons according to the Rayleigh law (Jensen, 2007). Both laws are wavelength dependant; however, aerosols by way of Mie scattering are notably less so (Jensen, 2007)

than Rayleigh scattering. The different scattering properties caused by the amount and type of optical interference therefore leads to dissimilarities in spectral responses of surface materials.

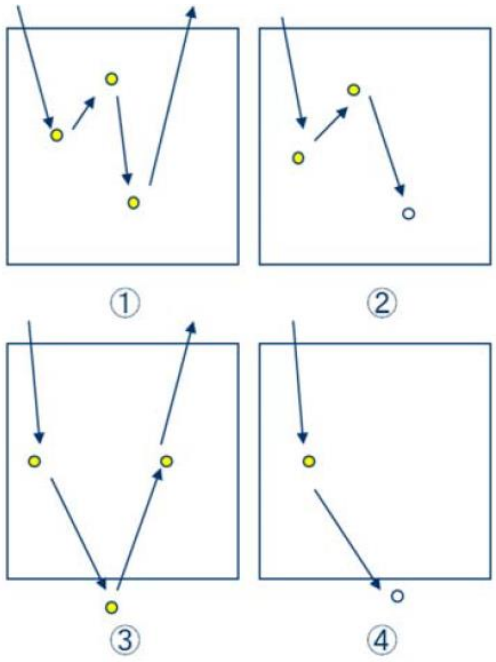

Figure 2.4: Four possible interactions with atmospheric molecules and aerosols (source: Aria, 2013).

The current and planned satellite based hyperspectral imagers have revisit times, with nadir imaging capabilities, ranging between 7-60 days (see table 1.1). Such coarse temporal resolution combined with only selecting imagery of optimal sun/sensor geometries may result in an increased probability of poor atmospheric conditions from available imagery. Less than ideal atmospheric conditions, therefore, may be unavoidable when working with such satellite based imagery.

2.2.3 Sub- And Pixel Level Landscape Spatial Structure

Classification of scene materials depends on the spatial structure of the landscape and the properties of its materials. The intimate nature of material interactions is complex as there can be an assortment of variations of material spectral signatures, intensities, spatial shapes, sizes, patterns, and intricate mixtures of each. Each of these relates to the spatial resolution of the imaging system and can be seen independently as a variable at the sub pixel and per pixel level (Woodcock et al., 1988; Smith et al., 2003; Neville et al., 2003).

Spectral properties of rock minerals

The spectral properties of scene materials with high absorption properties (darker materials) both, reduce the SNR (Nieke et al., 2000) and will be lost to the dominance of higher albedo materials shared within pixels (Gupta et al., 2000). The probability that a pixel will be assigned to a class associated with high albedo thereby increases if the pixels in which they lie are shared by materials characterized by lower albedo. Related to this effect and important to geological mapping is that mineral grain size and packing properties influence wavelength position absorption depths: as the grain size decreases, less light is absorbed and the reflectance increases (Clark, 1999). Rocks with finer grain properties therefore increase the probability of detection in mixed environments.

Background and targeted spectral differences

The differences between map units and background spectra (such as vegetation) also influence the ability to classify units. In areas with significant spectral differences between targeted and background the detection is less challenging.

The classification of rock mineralogy properties relies on very fixed located absorption features or spectral shapes. When a pixel containing specific rock spectral properties is mixed with other landscape material the spectra are blended for that pixel. If the spectra of the two materials contain absorption features or shapes that overlap or the blended spectra incurs a muted and altered shape they become more challenging to delineate. Spectral signatures of minerals will be compromised by the reflectance of the shared materials for the pixel. The numerous possible variances of multiple materials shared within pixels increases the complexity of the blended spectral signature. Landscapes with higher numbers of materials increase the chance of mixing and the difficulty of resolving their interaction.

Variance of targets and background

The spectra of rock minerals, although each uniquely different, can at times have varied spectra that is coincident with either other minerals or background materials. For example, rock outcrops may variably weather or contain differences in grain and packing sizes within a scene, and as a result increase the variability of possible extracted spectra. Mineral optical density properties also influence the variability in spectra as the allowable light that penetrates through a mineral rich rock will reflect other surrounding rock materials (Gaffey, 1996). Such dissimilarities in the spectra of minerals produce alterations in their shape and decrease the feasibility of accurate classification. In contrast, materials that are dominated by homogenous spectra or imagery that has been filtered to decrease the variance of material spectra are known to increase classification accuracies (Cushnie, 1987; Wulder et al., 2004).

Relationships between landscape materials also play a role in the spectral variability of representative rock unit spectra. Lichen for example, not only obscures mineralogy in variable degrees, (Ager and Milton, 1987; Zhang et al., 2005) it also physically and chemically weathers minerals and thereby alters the mineral spectra of units (Chen et al., 2000).

Landscape spatial structure patterns

With increased landscape heterogeneity and fragmentation, classification accuracies will decrease (Smith et al., 2002, 2003; Lin et al. 2008). The patch size of the landscape materials also incurs an effect on accuracy, however the relationship between patch size and classification accuracies compared to landscape heterogeneity has been found to be weaker (Lin et al. 2008). The significance of these negative spatial structures and their effects on accuracy are known to be material dependent. Some features can be strongly influenced by heterogeneity and not patch size and vice versa (Smith et al., 2002, 2003). Regardless, both heterogeneity and patchiness of a landscape decrease classification accuracies, and increase the need for higher spatial resolution to differentiate unit classes (Smith et al., 2002, 2003). These influences are stronger at sub-pixel than at the pixel level (Lin et al. 2008) and thereby impose challenges for coarser resolution datasets.

2.3 Sensor Specific Considerations

Remote sensing spectrometers are characterized by several different system performance metrics. Each of these metrics is determined by technological and physical limitations during the time of launch and a sensor's degradation through time. Many of the

technological limitations are determined by the volume, mass, and power requirements of the satellite launch. These limits are set by available satellite and sensor development funding (Villafranca et al., 2011). System design not only considers compromises between the cost of performance improvements but also the trade off in the improvement in one parameter at the cost of another all while considering the intended operational usage of the sensor.

2.3.1 Spatial Resolution

Sensor Design

Spatial resolution is, in this study, the nominally circular ground sampling distance measured by the sensor that is resampled to represent rectangular pixels in a digital image. The size of the aperture, focal length, altitude of the sensor, and dwell time govern the spatial resolution of an image (Kerekes and Landgrebe, 1989). The latter of these is controlled by the platform motion, stability, type of sensor, and onboard image processing capabilities of the sensor (Villafranca et al., 2011).

The circular spatial response measured by the sensor, termed the Point Spread Function (PSF) is nominally Gaussian in shape for most optical spectrometers (Kerekes and Landgrebe, 1989). The resampling of the PSF to derive rectangular pixels is typically done at the Full Width at Half Maximum (FWHM) of the signal. FWHM values are used as the PSF is assumed to be imperfect due to the motion of the satellite, the optical system properties, and the influence of the atmosphere on the spectral signal of any given pixel (Townshend, 1981; Coppo et al., 2013). The spectral signal thereby contains a

degree of reflectance from the surrounding area of a pixel and in some cases, by as much as three times the area of the reported spatial resolution of the sensor (Townshend, 1981).

Although the PSF response is considered nominally circular, it can be different in shape in both the x (across track) and y (along track) directions determined by the sensor's optics and platform motion (Kekeres and Landgrebe, 1989). With the Landsat ETM+ sensor there is a noted difference in these directions with the down-track response being asymmetrical and the across track response being symmetrical (Kavzoglu, 2010). Additionally, the size and shape of the PSF may not be consistent across spectral bands (Schläpfer et al., 2007). Further non-uniformity exists, according to Kavzoglu (2010), due to changes in the PSF during the launch of the satellite and with time due to out-gassing in the sensor system.

Classification influences

Spatial resolution is significantly linked to classification accuracies as it contributes to a major factor of information contained within imagery. Investigations into the optimal resolution for remote sensing studies have found the ideal resolution considers the factors discussed in section 2.1, namely the size, shape, optical influence, spatial heterogeneity and spectral diversity of both the background and the targeted features as well as the amount of imagery noise (Irons et al., 1985; Woodcock and Strahler, 1987; Hsieh et al., 2001; Chen et al., 2004; Ming et al., 2010).

Several assessments have been made on the effect of changing the spatial resolution of imagery (Tobler, 1969; Wiens, 1989; Turner et al., 1989). From these studies, several

observations have been made concerning how the content of data changes with scale. These include: (a) the standard deviation, and to a lesser extent, mean values of an image scene decrease with coarsening spatial resolution (Moody and Woodcock 1999; Gupta et al, 2000; Hubert et al., 2012), (b) low reflectance objects decrease in image percentage when upscaling (Gupta et al. 2000), (c) the detection of smaller objects decreases when the pixel size decreases beyond the size of the objects (Staenz et al., 2001; Behnia et al., 2012), and (d) spatial patterns and spectrally (dis)similar land cover types control the changes in class/ material percentages (Justice et al.,1989; Moody and Woodcock, 1995).

Also, materials with homogeneous spectral properties are known to be less sensitive to spatial resolution changes whereas materials with higher internal variance of spectra necessitate coarser spatial resolution imagery (Cushnie, 1987). Too coarse of a resolution however leads to pixels that contain spectra consisting of multiple landscape materials. Conversely, choosing too fine a resolution leads to various disadvantages. These include imagery that: a) gains spectral variance in targeted and background material, b) narrows the possible swath coverage for a given instrument's field-of-view (FOV), c) increases data volume, and d) increases demands on geometric correction requirements (Cushnie, 1987; Barnsley and Kay, 1990).

A generally accepted method of determining the optimal resolution for a given study is through the use of the geostatistical method called 'semivariogram analysis' (Balaguer-Beser et al., 2013). It is used to define the scale and pattern of spatial variability (Curran, 1988; Woodcock et al., 1988). The inherent maximum spatial dependence of the landscape with distance is determined by the location of the range (maximum curvature)

in the semi-variogram. Half or finer of this range distance is considered to be the appropriate spatial resolution (Rahman et al., 2002).

Geological mineralogy and lithological studies looking at the influence of spatial resolution find with changes in resolution extracted endmembers retain their spectral absorption features but exhibit changes in their amplitudes (Staenz et al., 2001 & 2005). The detection and classification of spatially small occurrences of rock minerals becomes limited with coarsening spatial resolution (Kruse, 2000; Staenz et al., 2001; Kruse et al., 2003, 2011).

2.3.2 Spectral resolution & wavelength coverage

Hyperspectral imaging spectrometers suitable to geoscience applications commonly measure the electromagnetic spectrum with a spectral coverage of roughly 400-2400 nm because a large number of typical minerals in study areas contain distinctive profiles and absorption features at specific wavelength positions in this range. These diagnostic characteristics can be used to identify their presence: fine spectral resolution allows for the detection and separation of some mineral that have characteristic narrower diagnostic features. The appropriate spectral resolution is therefore dependant on the mineral composition of lithologies of the region and the spectral absorption features of these lithologies.

Designing an imaging system with an optimally fine resolution however is technologically difficult. The choice of spectral selection elements in the system affects spectral overlap between bands, what provides the continuous coverage of the recorded

spectrum. The band widths are typically defined by a FWHM value of the measured signal. Additionally, the spectral response of each band requires measurement of a wide range of signal intensities (dynamic range) and if high intensity signals in one band are measured next to lower intensity measurements in neighbouring bands, band optical cross-talk can occur. The challenge of reducing this cross-talk however increases with attempts to improve the spatial and spectral resolution (Jerram et al., 2010).

When dealing with coarser spectral resolution imagery the position of bands is critical to rock mineral detection/separation. Band position can ensure that bands are centered on material spectral absorption features or conversely may be offset sufficiently such that two bands contain the absorption feature and reduce their depth. The result is a compromised ability to separate closely related spectra.

Spectral resolution and the quality of the spectral response function is a trade-off between spatial resolution, noise requirements, cost, onboard data storage limits, and optics. Any given spectrometer thus has varied responses to the quality of its spectral resolution. Disappointingly, no known studies were found that considered the effect of decreasing spectral resolutions with regards to mineral or lithological detection levels. However distinguishing features associated with typical minerals have narrow features particularly in the SWIR of approximately 10 nm. Figure 2.5 shows the spectral signature of the mineral kaolinite where in the 2.1 to 2.25 μm wavelength range contains its characteristic doublet absorption feature that is lost beyond resolutions coarser than 10 nm. Several minerals have absorption features in this range but without the doublet including halloysite and dickite (Clark, 1999). Without the 10 nm spectral resolution the detection

of this mineral would vary and add a degree of uncertainty in the ability to derive mineral and material abundances from imagery.

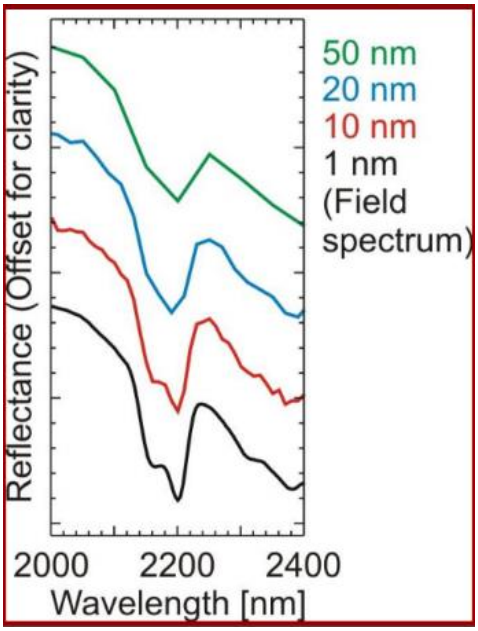

Figure 2.5: Continuum removed spectral signatures for the mineral kaolinite with decreasing spectral resolutions. Note the loss of the doublet absorption feature at 2180 nm (source: Kaufmann et al., 2011).

2.3.3 Radiometric resolution

Quantization or radiometric resolution refers to the precision of the recorded radiance measurement of an imaging system. It is determined by the conversion of the array's analog signal to a digital recording. During this process, pixel values are coded in a range of possible brightness values and stored as digital numbers (DN). The measure of this quantification is determined by the system's bit depth capability of its analog–to-digital converter (Jensen, 2007).

Pixel bit depth on several multispectral missions including Landsat 7 and ASTER are limited to 8-bit whereas planned hyperspectral satellites, PRecursore IperSpettrale della Missione Applicativa (PRISMA) and EnMAP have expected pixel depths of 12 and 14 respectively (Lopinto & Ananasso, 2013; Gaunter et al., 2015). This value determines the possible range in measurement where each pixel can have a value of 0-1023 in the case of 12 bit depth and 0-16,383 in the case of 14 bit depth.

The logic that there would be an improvement in mapping capability with increased radiometric resolution however has been found to be minor and questioned as to its added benefit. Harris et al. (2014) found that classification accuracies of bedrock lithological units from Landsat 7 and 8 (8 and 14 bit) were only minor (2.1% when comparing raw band data). Legleiter et al. (2002) similarly found that 11-bit compared to 8-bit PROBE-1 hyperspectral imagery during land cover classification only improved class accuracy by 0.8-2.1%. Irons et al., (1985) also looked at quantization differences. Landsat MSS data at 6 and 8 bits showed a consistent increase of just 3% when comparing land-cover class accuracies. Additional data storage and computational power is also needed to process this higher radiometric resolution data.

2.3.4 Image Quality

Image quality is largely defined by the SNR and dynamic range (DR) of an imaging system (Wang et al., 2012). The SNR is the more commonly used metric defining the image quality. It refers to the ratio of the detected signal to the noise created by the sensor. Higher values refer to less imagery noise and improved image quality. Noise

affects the shape of spectra. Since remote sensing conventionally identifies scene materials by comparing derived image endmember spectra to spectral libraries through regression analysis, noise is a deterrent to identification of scene materials.

Imagery noise comes from many sources in a spectrometer system but broadly can be divided into two categories: random and fixed-pattern noise (Acito, 2011). Random noise consists of photon (shot) and thermal noise sources whereas fixed-pattern noises are incorrect signals that have a constant behaviour within the imagery. Sources of fixed pattern noise include: imperfect calibrations of the detector, non-uniformity in the detector array, dead and bad pixels, and scratches (Swayze et al., 2003; Theuwissen, 2008; Acito, 2011). When a fixed pattern of noise in the along-track direction shows as lines in an image it is termed stripe noise. Random noise refers to the statistical variation in the number of photons detected by the sensor during array detection. The removal of random noise through post-processing is relatively more complex than the removal of fixed pattern noise as the latter are either known before launch, detected as a failure after launch, or can be sufficiently detected with algorithms (Acito, 2011).

Dynamic range, the other defining measure of image quality, refers to the ratio of minimum to maximum measurable saturation levels in detectable electrons. In other words, it is the contrast capability of the sensor. A sensor with the capability to detect wide variations in illumination has benefits when imaging a scene with varied surface intensities. The difference between natural environments and sensor capabilities thereby becomes a factor when attempting to derive information from scenes with bright environments (such as snow) and features with strong absorption properties (such as clear

water). Natural environments have a DR of roughly 100 dB, whereas most current satellites have a DR of 60-70 dB (Gamel and Eltoukhy, 2005). It can be seen then that increased DR is of significance to detecting targets in spectrally varied environments.

2.3.5 Sensor Type

Of the components that contribute to the overall performance of imaging spectrometers, the most significant in determining the overall performance is the sensor array (Gamal and Eltoukhy, 2005). Three array types are currently used in visible and short wave infrared wavelengths system designs: Charge-coupled devices (CCD), Complementary metal-oxide semiconductors (CMOS), and to a lesser extent mercury-cadmium-telluride (HgCdTe). Since the 1970's, the beginning of digital imaging, CCDs have been dominantly used in satellite imaging systems. They traditionally have higher pixel counts and thus higher spatial resolution (Rogalski, 2012). CMOS on the other hand have been brought on by the need for low cost and power requirements typical of many high volume handheld devices on the market today (Gamal and Eltoukhy, 2005). And lastly, HgCdTe sensor technology is the current leading edge performance sensor for infrared measurements and is being brought on by the needs of the latest astronomical instruments (Qian, 2015).

Sensor types lead to accuracy differences because there are imaging advantages and disadvantages of each. CCDs have an exceptional ability to measure a high ratio of the photons that reach the sensor material in the optical wavelengths, termed Quantum Efficiency (QE). However, because the pixels of these arrays convert a charged analog

signal to a digital signal to be sent off to be further processed, exposure times are much longer and power requirements higher as compared to CMOS and HgCdTe arrays (Gamal and Eltoukhy, 2005). On CMOS and HgCdTe sensors the conversion of the analog to digital signal takes place at the pixel level, and results in lower bandwidth requirements during read out compared to CCD's analog signal outputs. Smaller band width of the output signal thus results in better SNR (Rogalski, 2012).

CCDs are susceptible to cross-talk of high intensity signals in one band influencing those that are less bright in another band (Jerram et al., 2009) whereas CMOS suffer from poor low illumination performance and pixel level nonuniformity due to their pixel level analog to digital conversion (Gamal and Eltoukhy, 2005). The disadvantage of CMOS is that the space required to convert the signal to digital is at the cost of available surface space for measuring charge transfer thereby reducing its fill factor (Gamal and Eltoukhy, 2005). HgCdTe sensors have the best performance of the three types within the short wave infrared wavelength range as they have the highest QE, lowest dark current and read-out noise (covered in the next chapter) and demand less power to cool the sensor (Beletic et al., 2008; Qian, 2015).

Dynamic range capabilities of the three types of arrays are another decisive performance difference. The DR of CCDs are roughly one and a half that of a CMOS sensor (Gamal and Eltoukhy, 2005). However the high speed read out and digital output from pixels in CMOS arrays (Tack et al., 2012) means that during acquisition they can take multiple exposures with different contrast settings (Gamal and Eltoukhy, 2005). By taking multiple captures high dynamic images can be created. High dynamic range images from

CMOS imagers can produce imagery with DR of 100 dB, higher than CCD's DR of 60-70 dB (Gamal and Eltoukhy, 2005).

Despite the wide use of CCDs on existing spectrometer satellites like MERIS, CHRIS, and Hyperion's VNIR spectrometer, CMOS type sensors are gaining more popularity in new scientific designs (Holst, 2009). Specim's, a hyperspectral manufacturer, latest airborne spectrometer called the AISA Fenix, the planned EnMap spaceborne sensor, and a new university research spaceborne experimental hyperspectral sensor (Praks et al., 2011) all use CMOS type arrays. Given that cost less, require less power, physically smaller, and less complex than standard CCD technology, future missions will increasingly use CMOS sensors (Theuwissen, 2008; Jerram et al., 2009; Rogalski, 2012).

2.3.6 Spectral and Spatial Distortions Keystone and Smile

Spectrometers suffer from two known non-uniform distortions of the sensor array: smile and keystone. They are known to negatively influence classification particularly when unmixing pixels (Dadon et al., 2010). Keystone is a spatial distortion that occurs in the spectral domain with the across track direction of a sensor array (figure 2.6). It is caused by changes in the magnification, caused by the instrument narrow opening, with wavelength (Mouroulis, 1999). The influence of this distortion varies with wavelength and spectrometer (Guanter, 2009). Keystone distortion is typically larger at shorter wavelengths (Guanter, 2009) and only significantly large enough to effect material detection within VNIR wavelengths (Dadon et al., 2010).

Smile is a spectral distortion in which the central wavelength position of bands changes with the spatial dimension of the sensor array (Yokoya et al., 2010) (figure 2.6). The effect is that spectral resolution decreases towards the outer edges of an image, and thus outer pixels in the along track direction can be considered to be biased compared to the central pixel in the array. (Ceamanos and Doute, 2009).

Although correction of both keystone and smile are frequently accomplished in post processing using pre-launch information, their specifications may likely change after their launch. This complicates their correction (Yokoya et al., 2010). In some cases these effects are simply ignored by analysts due to their complexities (Dadon et al., 2010).

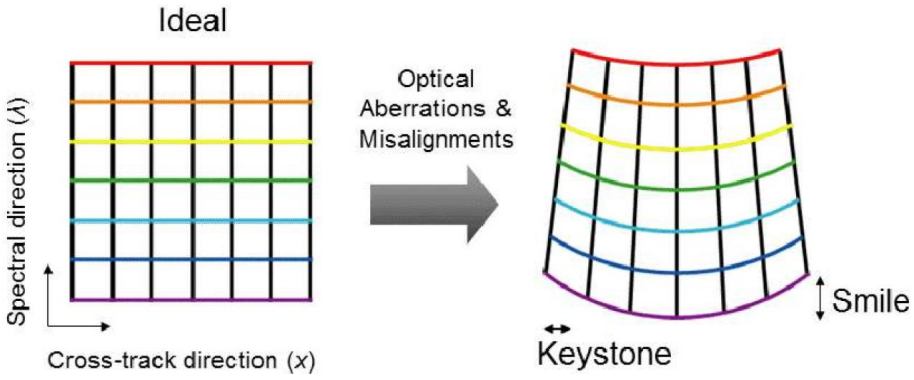

Figure 2.6: Keystone and smile (Source: Yokoya et al., 2010).

2.6: Dual Spectrometer Co-Registration

Remote sensing hyperspectral satellites measuring both the VNIR and SWIR utilize two separate overlapping spectrometers: one for each of these spectral wavelength ranges. These individual spectrometer designs are optimized for their spectral coverage based on different exposure times and array material requirements. Silicon based arrays are

dominantly used for the VNIR wavelengths and several various possible materials are used for the SWIR wavelengths (Rogalski, 2012). Because of these structural differences a small-time delay exists between spectrometer measurements. This delay results in slightly different viewing angles which leads to a spatial difference between sensors (Schwind et al., 2012). The co-registration of the imagery thereby leads to spatial errors. These errors are more pronounced in imaged locations with steeper terrains. Schwind et al. (2012) found that a displacement error of 11.44 m occurs on 45° slope surfaces compared to flat surfaces with 8.13 m displacements. Geometric correction of this co-registration error is therefore dependent on sensor design, satellite altitude, and the topography of the imaged scene.

2.3.8 Scene Size Capability

Scene size is influential in the detection of scene targets beyond the practical implication of a larger spatial coverage for a given study area. The scene size is also influential in increasing the probability that pure pixels can be discovered. With more landscape coverage there is improved capability to derive well representative endmembers from a scene: a challenge for bedrock geology mapping in Canada's Arctic. However, sensors are challenged by limited data storage, compression, and down link capabilities that can limit their field-of-view (FOV). Additionally, if the sensor FOV is sufficiently large then the off-angle view at the outer edges of the image will lead to spatial errors as well as changes in the ground sampling distance. This results in a decrease of the sensor's spatial resolution (Segl et al., 2012).

Similar to the effect of larger FOV degrading the spatial resolution, any increased off-angle pointing view of the satellite will again influence the spatial sampling distance and thus impact the spatial resolution (Segl et al., 2012). Some satellites grant an across track look-angle up to 30° whereas others are fixed at nadir. The degraded ground sampling distance of off-angle views is important to note when comparing sensor datasets as the reported spatial resolution is dependent on the sensor's acquisition parameters.

Off-angle views are additionally plagued by another negative impact on their imagery: light is refracted at an angle due to the Earth's atmosphere and causes geometric errors (Noerdlinger, 1999). This effect can lead to ground displacement of pixels roughly 2 m at 30° off-nadir view angles for the EnMap sensor (Schwind et al., 2012). Although 2 m may not be important when considering a 30 m size pixel it does however highlight the challenge of faithfully capturing the scene in a uniform grid.

2.4 Processing Considerations

Although the intent of this chapter is to determine the factors that control mineralogical detection that pertain to the sensor and acquisition parameters for a given scene, there are several other influential factors worthy of briefly discussing. Several critical steps are conventionally accomplished after the image is acquired that effect derived abundance and classification maps.

Differences in atmospheric correction methods can change the results of detection algorithms (San and Suzen, 2010). The application of accurate and sufficient ground control points (GCPs) during geometric correction of the imagery is necessary and

particularly important in high relief imaged regions. Insufficient, or poor GCP location or distribution, can lead to spatial errors and thereby topographic correction and ground truth data matching becomes flawed. Classification and derived abundance maps are produced from a considerable number of possible approaches including pixel based and pixel unmixing algorithms. The algorithm to be used however is reliant on accurate representative spectra for targeted classes / materials (Lu and Weng, 2007). These are influenced by the endmember extraction method used, or in the case of pixel based classification, the training data's size, number (Gong and Howarth, 1990), and sampling design (Congalton and Green, 2008).

2.5 Conclusion

This chapter has provided a brief overview of a number of variables that influence the classification of rock lithologies and the identification of their mineral compositions. Not only do sensor technology limitations play an important role, but also the scene properties, the acquisition parameters, processing steps, and the atmosphere. These factors are important to acknowledge when comparing sensors. Understanding the effect that difference in system parameters have is critical. Attributing statistical differences between results to the sensor capability can be flawed if attempts are not made to match similar geometries, illumination, and processing steps.

The information provided in this chapter also helps to outline the considerations required when simulating imagery of expected new satellites. During simulation studies it is optimal that the input data used for modeling be of high spatial and spectral resolution (Zhao et al., 2010). It is therefore clear that chapter that removing the effects of the

atmosphere, spectral influences of the landscape materials, sun/sensor/surface geometries and the acquisition sensor's limitations prior to modeling the sensor under assessment is ideal. This process is referred to as 'backward modeling' and produces a high spectral/spatial resolution image, or 'superresolution' image, of the study area (Kavzoglu, 2004; Zhao et al., 2010; Segl et al., 2012). Forward modeling, the next step in the modeling, uses this super-resolution imagery to incorporate all the known sensor parameters under assessment.

If the information provided in this chapter were to be used to help with the selection of the appropriate sensor or acquisition parameters then the considerations of desired scale, landscape, and targeted map units thereby forms an integral part of this decision.

Chapter 3: Simulation of EnMAP Data through Upscaling of Airborne HSI Data

3.1 Introduction

With the expected launch of several hyperspectral satellites in the next decade, including EnMap, PRISMA, HyspIRI, and HISUI, there is interest in assessing their capabilities (Buckingham and Staenz, 2008; Staenz & Held, 2012). Geologists and scientists from other disciplines are eager to test and develop improved processes for these and other sensors in anticipation of their use. Simulating the imagery obtained from these satellites can help better understand their predictive performance, and applicability to a problem (Bian & Butler, 1999; Börner et al., 2001; Staenz et al., 2001, 2005; Peddle et al., 2008; White et al., 2010; Kruse et al., 2011; Segl et al., 2012; Rogge et al., 2014). It can also reduce expenditures during remote sensing aided mapping as it helps determine the most appropriate image sensor prior to any satellite data acquisition. Savings can likewise be made for design engineers that are considering compromises in system designs that should be made due to technological requirements and for assessing the cost of quality improvements (Kerekes and Landgrebe, 1989; Börner et al, 2001; Coppo et al., 2013). And finally, remote sensing practitioners can use simulated data to assess optimal preprocessing and classification algorithms under varying sensor parameters (Börner et al, 2001; Segl et al., 2009; Liu et al., 2009).

It is not surprising to learn that several spaceborne hyperspectral scene simulation tools have been developed to address these justifications. Some tools are sensor specific and as such are highly accurate to their modelled sensors (EnMap: Segl et al., 2009, PRISMA:

Barducci et al. 2010; HISUI: Matsunaga et al., 2011) others are open-ended to incorporate variable sensor parameters (SENSOR: Börner et al., 2001). Simulation tools are also often intended for specific phases of spaceborne satellite missions. Some are intended for the conceptual and preliminary analysis of systems (Phases 0/A) (Blonski, 1999) whereas others focus on defining system performance requirements (Phase B)) and system design and development (Phase C) (SENSOR: Börner et al., 2001; EnMap: Segl et al., 2009; PRISMA: Barducci et al. 2010; PICASSO: Cota et al., 2010; Selex-ES: Coppo et al., 2013).

Regardless of their intent, these tools all have challenges that limit their use. The complexities of these simulation scripts require expertise by either the scientists that developed them, or necessitate training that is limited to specific nationals (Digital Imaging and Remote Sensing Image Generation (DIRSIG), 2016). Additionally, available tools are often written in programming languages that utilize commercial software as part of their operation (Kerekes, 2002; Kerekes & Buam, 2003; Segl et al., 2012; Coppo et al., 2013).

This chapter outlines the theoretical framework and rationale for the simulation of a hyperspectral satellite image. This framework was used for the creation a series of scripts written in an open-source programming language to simulate six key hyperspectral sensor parameters. The Comprehensive R Archive Network (CRAN) for the R programming language (version 3.2.2) was chosen for this thesis because of its relative ease of use and free access that would enable anyone with a basic knowledge of this common programming language to alter the variables for creating their own simulated scene and to

ensure its ongoing use. The scripts simply require entering the desired parameters and an airborne spatially and spectrally oversampled reflectance imagery dataset for their input.

Six key image scene parameters are modelled in the scripts including sensor jitter and motion blur, spatial resolution, spectral separation (where dual spectrometers are modelled), spectral resolution, and signal dependant and independent sensor noise. The scripts work in addressing each of these separately and in this order. Figure 3.1 outlines this workflow. Five scripts make up the simulation and the content of each are noted in the workflow diagram and are noted at the beginning (top) of each script. All the parameters for the scripts are entered in the first script and the remaining four scripts do not require the user to modify; these remaining scripts are called for during the running of this first script.

Five datacubes are outputted during the simulation: a spatial convolved, a spectral convolved, a total additive noise, a final simulated datacube, and this same datacube but without the overlapping SWIR bands (900-1000 nm). The last two datacubes are formed from the preceding three treatments.

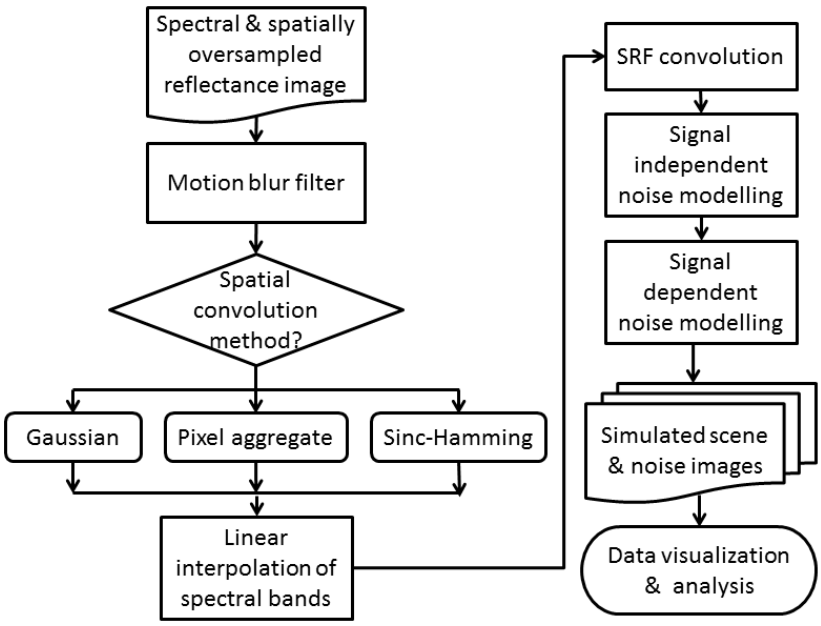

Figure 3.1: General processing workflow

Several parameters and considerations are not part of the modelling scripts in this thesis that maybe seen in other end-to-end simulation tools but are outside the scope of this thesis. Seven considerations could additionally be modelled: a) the modelling of atmospheric absorption due to an increased optical depth of the satellite altitude, b) backward modelling of sun, sensor, and surface geometries during acquisition, c) radiometric (bit depth) degradation, d) co-registration errors in dual spectrometer systems, e) image compression for data downlink, f) variable spectral response functions (SRF) with changing wavelength, and g) the pushbroom spectrometer optical aberrations: keystone and smile.

This high number of considerations and the complexities of simulating the key parameters highlight the challenges necessary in developing simulation tools. A large number of studies conducting performance testing of planned satellites commonly limit

the simulation to convolving spectral and spatial resolutions and in some cases adding noise as these are potentially the most deterministic factors of their performance (Staenz et al., 2005; Peddle et al., 2008; Kruse et al., 2011; Kruse and Perry, 2013). In most cases these simulations, in an effort to save time take a simplistic approach to modelling as they don't consider sensor point spread function (PSF), spectral response function (SRF), and/or noise to be signal dependent.

3.2 Required Input Data

The scripts require a spatially and spectrally oversampled reflectance cube. The dataset prior to input is geometrically corrected to compensate for aircraft pitch, roll, and yaw movements during acquisition, and if necessary orthorectified to account for terrain and optical distortions. The input high resolution airborne data (most likely) requires multiple flight lines in order to build a close representation to that of the simulated sensor's swath and thus the flight line imagery requires radiometric balancing to create a single dataset.

Significant noise is removed when both spectrally and spatially convolving the data as this process filters the data (Börner et al., 2001; Meola et al., 2011). The data however must be of high SNR as any noise not filtered will contribute to lowering the SNR of the output data and degrade the quality of modelling.

In order to spectrally convolve the data there needs to be a greater spectral range of both the minimum and maximum wavelength in the input data than the desired output. This overlap requires being greater than half the desired output spectral resolution.

The script is written to convolve the spatial dimension as a ratio of the input to desired output of either 10:1 or 15:1 only. This limitation can however be extended to include other ratios by straightforward modification of sections of the code.

3.4 Spatial Convolution

The defined spatial resolution of a specific sensor is largely defined by its Ground-projected Instantaneous Field-Of-View (GIFOV). The GIFOV is determined by the sensor's Field-Of-View (FOV), number of pixels per image line, and its altitude from the imaged surface (Oppelt and Mauser, 2007). Since airborne data acquisition is conducted at relatively low altitudes they are higher in spatial resolution compared to satellite data. In order to convolve high resolution data to the satellite resolution the satellite sensors' Point Spread Function (PSF) has to be considered. The PSF describes how a system converts received signals to produce a pixel measurement. Signal processing theory dictates that ideally the PSF follows a circular 3D sinc function in shape (Markham, 1985; Smith, 1999). The scaling of the sinc window function is determined by ensuring that the Full Width Half Maximum (FWHM) is equal to the Ground Sampling Distance (GSD) or in other words the spatial resolution of the sensor. The sinc window function is defined by the following equation:

$$Sinc(x) = \frac{\sin\left(\frac{\pi}{f}x\right)}{\frac{\pi}{f}x}$$

Equation 3.1: The sinc window function

Where x is the position and f is the scaling factor of the function. The scale factor is determined as the value necessary to provide a distribution with a FWHM at that equals the sensor's GIFOV. Figures 3.2 and 3.3 illustrate in 2 and 3 dimensions respectively, how the reflected surfaces within the ground sampling area are not weighted equally using this function: pixels are heavily influenced by surface reflectance from the central part of the pixel, they incorporate signal outside the area of their pixel dimensions, and have a negative ringing response to some of the signals outside the pixel space. The result is a pollution from neighbouring pixels/signals which blurs edges and rounds the shapes of objects (Bøcher and McCloy, 2006).

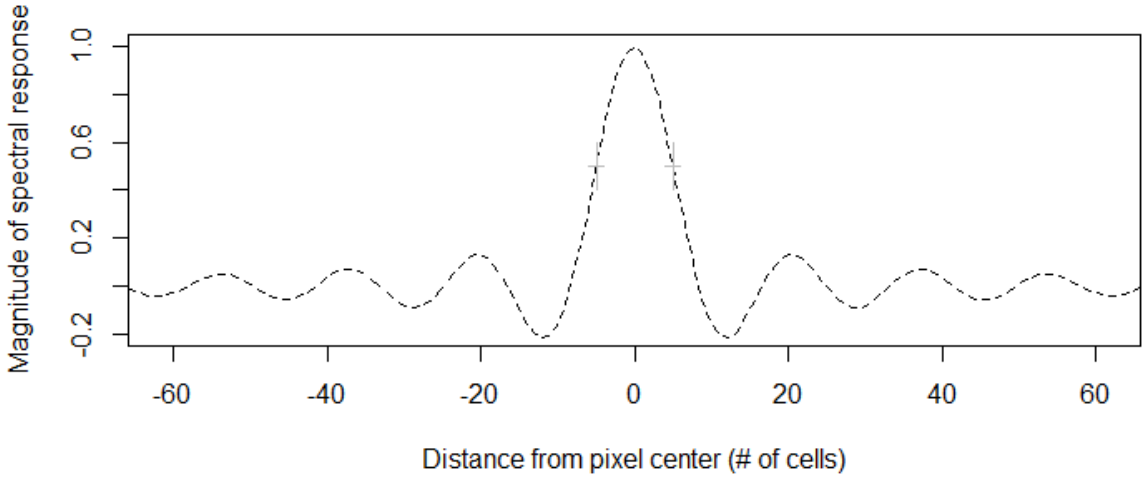

Figure 3.2: A 2-dimensional plot of the sinc function for a PSF with a FWHM equal to 10 pixels of the input data. Observe the infinite outward lobes. Grey crosses mark the GIFOV and FWHM.

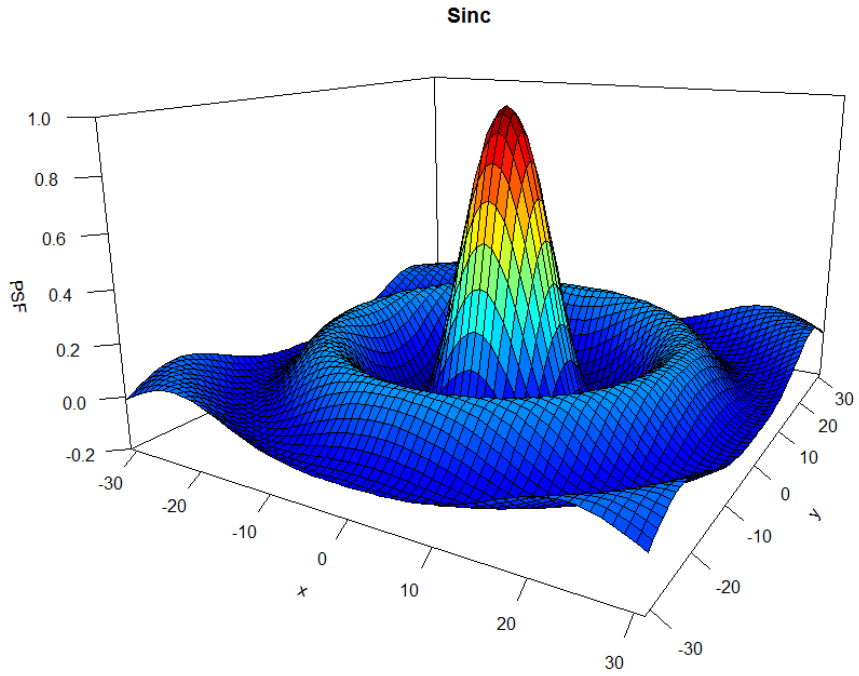

Figure 3.3: A 3-dimensional plot of the sinc function demonstrating the negative (Y-axis) influence of the signal with the outward lobes.

Ideally the PSF for a given sensor is known since it can be measured and can be employed for simulation yet it often is not known, or is assumed to be roughly a sinc

window shape. Simulating using the sinc function response however is impractical as the response slowly infinitely decays and never outwardly reaches zero. Simulation thereby pragmatically cannot be achieved digitally and only if the values are truncated at an outward limited extent (Smith, 1999). As a result most simulation tools consider the PSF as an isotropic Gaussian function (equation 3.2 and seen in figures 3.4 and 3.5) (Kerekes and Landgrebe, 1989; Kavzoglu 2004; Coppo et al., 2013). In spite of its similarity in shape and relative ease of use, this treatment produces an overly optimistic simulation of the imagery as it does not treat some of the negative response with the outward side lobes of the sinc function (Kerekes & Landgrebe, 1989), nor does it consider that the PSF might be asymmetrical in shape due to the motion of the sensor in the along track direction (Kavzoglu, 2004).

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Equation 3.2: The Gaussian window function

Where σ is the standard deviation of the pixel values and *x* and *y* the spatial position within the filter kernel window.

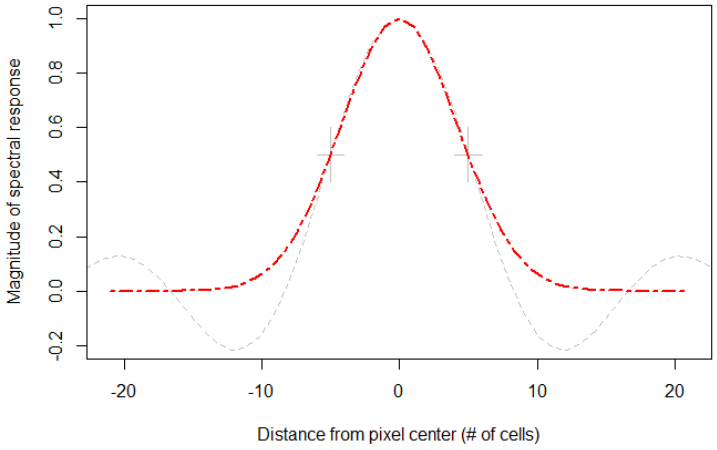

Figure 3.4: A 2-dimensional plot of the Gaussian function for a PSF with a FWHM equal to 10 pixels of the input data. The location of the FWHM (determining the spatial resolution of the simulated sensor) is shown in light grey crosses. The ideal sinc PSF is shown in the light grey dashed line for comparison.

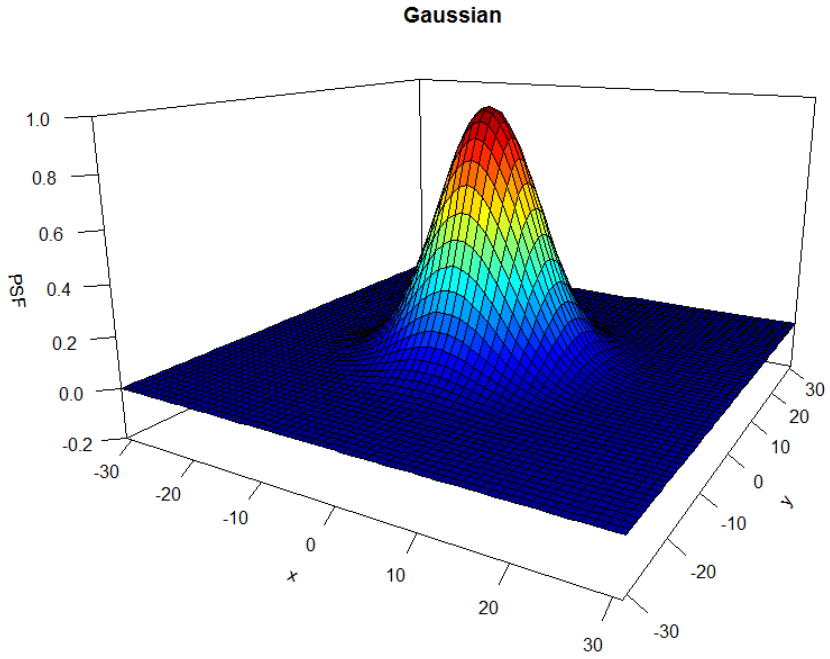

Figure 3.5: A 3-dimensional plot of the Gaussian function.

One solution to a more accurate representation of the PSF than the Gaussian window function kernel is proposed by Smith (1999). He suggests that multiplying the sinc

window function (equation 3.1) by a hamming window function (equation 3.3) provides a sensible alternative. The result (figures 3.6 and 3.7) is a window kernel that's general shape is similar to the sinc function, albeit without continuous side lobes and has a diminished central influence, but the PSF response reaches zero at roughly the same distance as a Gaussian function. By having the response reach zero it, unlike the sinc function, can now be applied to imagery to produce a more realistic simulation of the PSF than the Gaussian function.

$H(x) = 0.54 - 0.46cos\frac{(2\pi)n}{N}\ 0 \leq n \leq N$ *Equation 3.3: The hamming window function*

where *N* is the window length and *n* the position along the length.

Pixel aggregation, a process that computes and assigns the mean value of pixels within the new coarser resolution dataset using mean values, is another pixel convolution method often applied in fundamental sensor performance studies (Bian & Butler, 1999; Staenz et al., 2001; Kruse et al., 2011). Despite its dissimilarity to an actual PSF as there is no weighting of input values, it is used because of its relatively low computation requirements and availability in most image processing programs. The window size is relative small and only requires consideration of the pixels that make up the coarser simulated pixel. This method has also been termed 'block averaging' (Justice et al., 1989; Staenz et al., 2001) and 'nonoverlapping averaging' (Hay et al., 1997; Niemann et al., 1997).

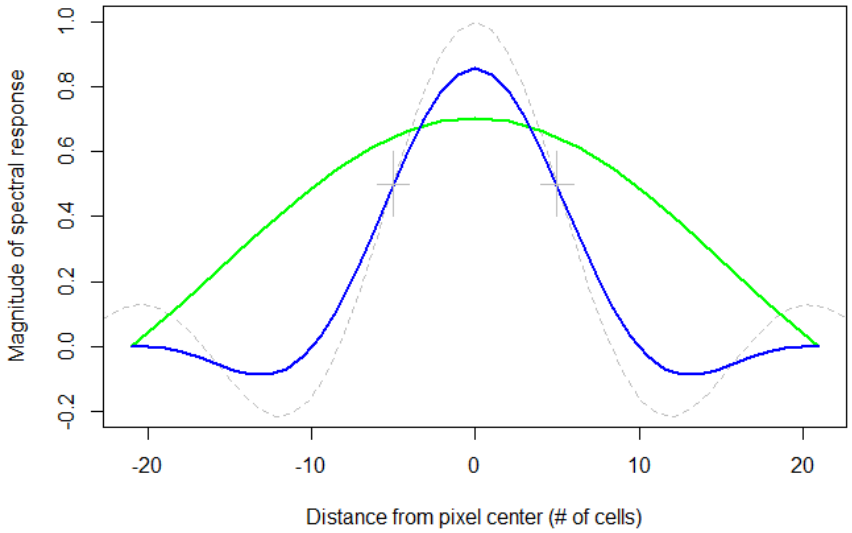

Figure 3.6: A 2-dimensional plot of the hamming window function shown in green and the result of multiplying this window by a sinc window function in blue. Note how the sinc-hamming window function terminates at 0 making it more practical to use than a sinc function terminates at 0 making it more practical to use than a sinc function.

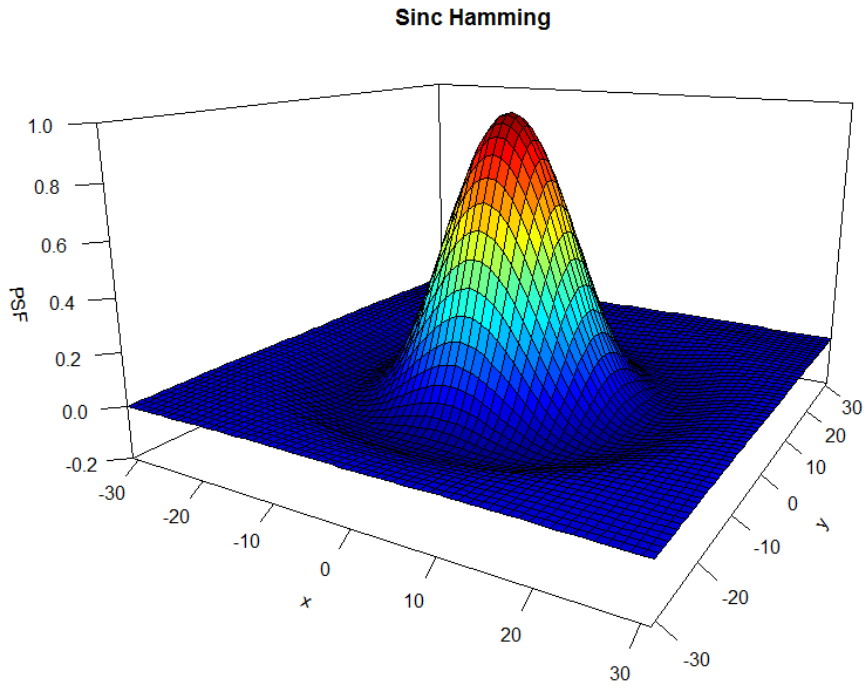

Figure 3.7: A 3-dimensional plot of the sinc-hamming function.

The CRAN R spatial convolution script enables the use of the three previously mentioned possible simulation PSF window kernel function: Gaussian, pixel aggregation (based on mean values), or a combined sinc-hamming kernel. These options are available as each demand differing computation requirements and thereby differing processing times. The combined sinc-hamming window kernel can assume to be the most realistic but has the longest running times, the Gaussian window kernel, mentioned elsewhere, is assumed to be only marginally less realistic but with slightly less processing time, and the pixel aggregation process takes substantially less time to run but with the least realistic modeling of the PSF. Both the Gaussian and sinc-hamming window kernels are sufficiently large to consider values till the weighting equals 0 and thus are non-truncated. The weighting of all three PSF window kernels are normalized to sum one. This ensures output pixel values are scaled appropriately.

Figures 3.4 through 3.7 show that the sinc-hamming and Gaussian window kernel functions consider pixel values from the high resolution input data outside of the simulated pixels dimensions. In order to construct simulated pixels along the edges of the imagery, additional imagery rows and columns need to be synthesized to deal with these PSF shoulders. The script handles this dilemma by mirroring the edges of the datacube at the necessary distance, automatically derived by the script.

During the process of spatial convolution using the moving filter the script first “pads” the edges of the imagery or in other words, the moving kernel window starts at an inward location within the data. This is necessary so that the window filter does not start at the edge of the data where there are no data to weight from outside the imagery edges.

3.5 Spectral Convolution

Spectral convolution is handled by first splitting the input data into two datasets based on spectral wavelength range. This is performed to simulate different Spectral Response Function (SRF) of dual spectrometers typical of satellite systems. Typically, dual spectrometers have differing spectral resolutions for each of the sensors: often narrower for the VNIR than the SWIR spectrometers. The script can also handle overlapping band ranges typical of dual spectrometer designs. The script considers any necessary extra wavelength necessary to convolve data to the desired range and resolution.

The SRF of documented hyperspectral imaging sensors is nominally considered to be a 2-dimensional Gaussian function. The scale of the function is determined by ensuring that the FWHM of the distribution matches the spectral sampling interval (SSI) which determines the spectral resolution. The central peak of the Gaussian distribution is centered on the desired spectral band position and is positioned to overlap the neighbouring spectral band (see figure 3.8). The spectral convolution process weights a 1 nm spectral resolution dataset considering the desired SRF. These 1 nm super-resolution data are derived by linear interpolation:

$$y = y1 + ((x - x1)*((y2 - y1)/(x2 - x1))) \qquad \textit{Equation 3.4}$$

where y is the spectral value at x the wavelength position. $y1$ and $y2$ are the two closest known wavelength positions above and below the desired position and $x1$ and $x2$ are the corresponding spectral values respectively.

Linear interpolation of the input spectrum is needed, unlike the spatial resolution, to compensate for the insufficient oversampling of the input data to adequately weight the input spectrum when modelling the SRF. The script achieves this interpolation regardless of the input spectral resolution however the spectral resolution of the output synthetic image is limited to 6.5, 7.5, and 10 nm as all known planned or existing spaceborne sensors have these resolutions. Other desired resolutions can be achieved with minor editing of the script.

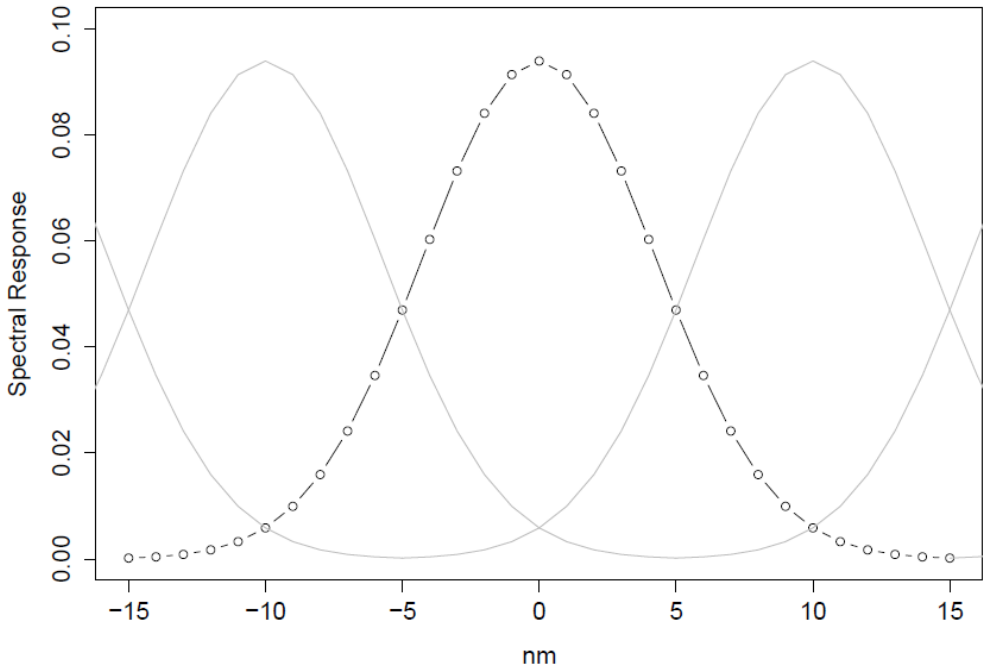

Figure 3.8: The Spectral Response Function of an individual 10 nm wide band plotted in 2-dimensions and shown in black. Grey lines represent the SRF of neighbouring bands. Circles represent weighted values used in the simulation at 1 nm intervals.

3.6 Noise

Noise is any unwanted component in the recorded signal of an image (Boncelet, 2005). It is the fluctuations within the spectra about a mean value and as result affects the sensitivity of the instrument. The presence of noise is an unavoidable part of imaging

technology and can only be reduced through either system design compromises or image pre-processing attempts.

Noise is typically measured as a ratio between the mean amplitude of the signal and the standard deviation of the noise. The quality of a signal is often a basic estimation expressed as the signal to noise ratio (SNR):

$$SNR = \frac{R\mu}{R\sigma} \qquad \textit{Equation 3.5}$$

where μ is the reflectance mean or expected value and σ is the standard deviation of the noise.

Noise originates from a number of sensor sources that are either dependent or independent of the measured signal (Nieke et al., 1999). Signal dependent noise is more prevalent in most illumination conditions (Alparone et al., 2009; Hobbs, 2011). The signal dependent component of noise is due to photon noise (sometimes referred as shot noise, Poisson noise, or Quantum noise (Hobbs, 2011)). Photon noise is caused by the measurement process in which there is an inherent statistical variation in the arrival rate of photons at the sensor array (Polder et al., 2003). Signal independent noise is the dominant source when there is low illumination within an image (Hobbs, 2011). It is caused by several sources including read-out, dark current (thermal or Johnson), salt and pepper (impulse), and quantisation noise.

Details of each of these sources of signal independent noise are as follows:

Read-out noise is the prevailing source of signal independent noise (Boncelet, 2005). It is a noise that arises by the electronic circuitry process converting the array pixels' charge to a digital/voltage signal. CMOS sensors have lower read-out noise because the conversion of the analog to digital signal, occurring within amplifiers, takes place at the pixel level rather than a slower process of each pixel's charges being passed to a separate analog signal amplifier as is the case in CCD type sensors (Theuwissen, 2008).

Dark (also termed dark current or thermal) noise occurs in imagery because of statistical variation in the motion of thermally generated electrons during the spectrometer dwell time. Its effect is considerably reduced by cooling of the instrument and increased dwell time (Kasap et al., 2009).

Quantization noise is the error that is induced when converting the sensed analog signal of the array to a digital signal but the digital recording system is not sufficiently developed to capture the subtle differences in the signal (Kasap et al., 2009). The result can be seen as a step like appearance over spatially gradual changes in values of an image (Boncelet, 2005). Quantization noise is considered to be negligible with current system designs given their level of radiometric resolution (Polder et al., 2003; Theuwissen, 2008; Chen, 2012).

Impulse noise refers to extreme value pixels that are a result of malfunctioning pixels in the sensor array, faulty memory locations, and during data downlink from the satellite (Chan et al., 2005). They appear as random isolated light or dark intensity pixels that give a salt-and-pepper look to imagery. Hence impulse noise is sometimes referred to as "salt-and-pepper noise".

3.6.1 Simulating noise

As the platform of a pushbroom sensor collects data, three detrimental effects on geometric fidelity occur as a result of platform instability in space: orbital motion, sinusoidal motion and jitter blur (Holst, 1998). The orbital motion of the sensor craft produces a directional smear in the imagery and varies depending on imaging dwell time. Sinusoidal motion and jitters are the high frequency random movements caused by onboard electronics. Sinusoidal motion and platform jitter result in blur in both the along and across track direction whereas motion blur only occurs in the along-track direction. Such blurring can be approximated with a Gaussian function (Holst, 1998).

The script applies a Gaussian shaped 3x1 and 3x3 windowed kernel filters to simulate the motion and jitter blurring respectively. This is accomplished as the first step in the simulation and is applied to the high resolution/fidelity imagery. The influence of the blurring can be controlled by a multiplicative factor parameter expressed as a percentage of the input signal. The blurring can also be turned off if not desired by setting this parameter value to 0.

Prelaunch sensor estimates of SNR levels can be used in the script and are entered separately for each spectrometer in the case of dual spectrometer designs. Band positions of the estimated SNR values are required for two positions within the instrument's measured wavelengths. As documented estimated SNR values for a sensor are for a specific albedo value (30% for EnMap), the script additionally requires this albedo value. The script linearly interpolates SNR values for the remaining band positions.

The simulated noise script artificially constructs both the signal dependent and independent sources of noise separately. The signal independent noise is first modelled followed by the signal dependent noise. This second contribution is calculated in the script by considering (removing) the signal independent noise from the instrument's desired SNR values.

The signal independent noise is generated by a series of random values following a Gaussian distribution with a zero mean. The user is required to enter the signal independent level of noise as a SNR value. This may be problematic given that the expected signal dependent noise contribution for any satellite sensor was not been found in any literature. In the absence of this information however this noise variance can be estimated (although optimistically) by considering the highest SNR for the highest albedo surfaces found in SNR graphs for the sensor. This value should reflect the highest achievable SNR and the upper limit imposed mostly by the signal dependent noise contribution. In the case of the EnMap sensor details occurs in Sang et al. (2008). The signal independent noise is thus applied to each band separately as:

$$Rsins_{ij} = \phi\left(0, \frac{minimum\ reflectance\ (Rin)}{SNR\ of\ miniumum\ reflectance\ (Rin)}\right) \qquad \textit{Equation 3.6}$$

where *Rsins* is the simulated signal independent noise, ϕ the Gaussian distribution with a *0* mean value, and standard deviation based on the mean (μ) value of the input reflectance image band (*Rin*) at each pixel position (*ij*). The script generates an image of zero mean Gaussian noise with the same extents and resolution as the input data with a standard

deviation of 1. This image is then scaled considering the stated desired SNR for the signal independent noise.

Signal dependent noise is also modelled as a random distribution of zero mean Gaussian values and its standard deviation is calculated as the square root of the measured signal strength (Nieke et al., 1999; Theuwissen, 2008; Kasap et al., 2009) applied sequentially to each wavelength band:

$$Rsdns_{ij} = \phi\left(0, \sqrt{Rin_{ij}}\right) \qquad \textit{Equation 3.7}$$

This zero mean Gaussian signal dependent noise requires scaling to achieve the desired SNR values for each band. Before scaling the required SNR for each band is calculated through linear interpolation of the entered VNIR and SWIR SNR levels (equation 3.4) as the SNR levels are not consistent across the measured wavelength positions (Segl et al., 2010). Once the determined SNR for each band is resolved the script calculates the SNR level of the signal *independent* noise and then calculates the required remaining noise for the signal *dependent* noise to achieve the desired SNR. This is calculated as the following:

$$scale_{ij} = \frac{\mu Rin}{(Desired\ SNR - \frac{\mu Rin}{\mu Rsdns})} \qquad \textit{Equation 3.8}$$

This scaling factor is applied by multiplication to the signal dependent noise:

$$Rsdns_{ij} = Rsdns_{ij} * scale_{ij} \qquad \textit{Equation 3.9}$$

The two noise contributions (*Rsdns* and *Rsins*) are added to the spatially and spectrally convolved reflectance cube to produce a simulated noisy image as follows:

$$Rsim = Rin + Rsins + Rsdns \qquad \textit{Equation 3.10}$$

Where *Rsim* is the desired simulated noisy reflectance image, *Rin* the reflectance image and *Rsins* and *Rsdns* are the signal independent and dependent noise of the pixel.

The results of each of the noise components are demonstrated in figure 3.9. This 3-D diagram shows the amount of noise the simulation applies dependent on the signal intensity and wavelength. Although noise values will randomly vary during any simulation, the values within this diagram are the average of four permutations at each intensity/wavelength position. Two surface planes are shown: the signal dependent (upper surface) and independent noise (lower). The figure shows how signal dependent noise is the dominant source of noise in most reflectance conditions and that the signal independent sources are of little significance particularly when the illuminance is intense. It can also be seen that when the reflectance is low in the short wavelengths (near 420 nm) the signal independent noise has a relatively larger contribution to the overall noise. It might initially seem to reader that since noise increases with more signal intensity that the SNR would lower with increased reflectance however since the noise is proportional to the square root of the signal the noise proportionally decreases with signal – thereby increasing the SNR. This leads to the outward curved shape along the reflectance axis for any given wavelength position – most notable at the higher wavelengths.

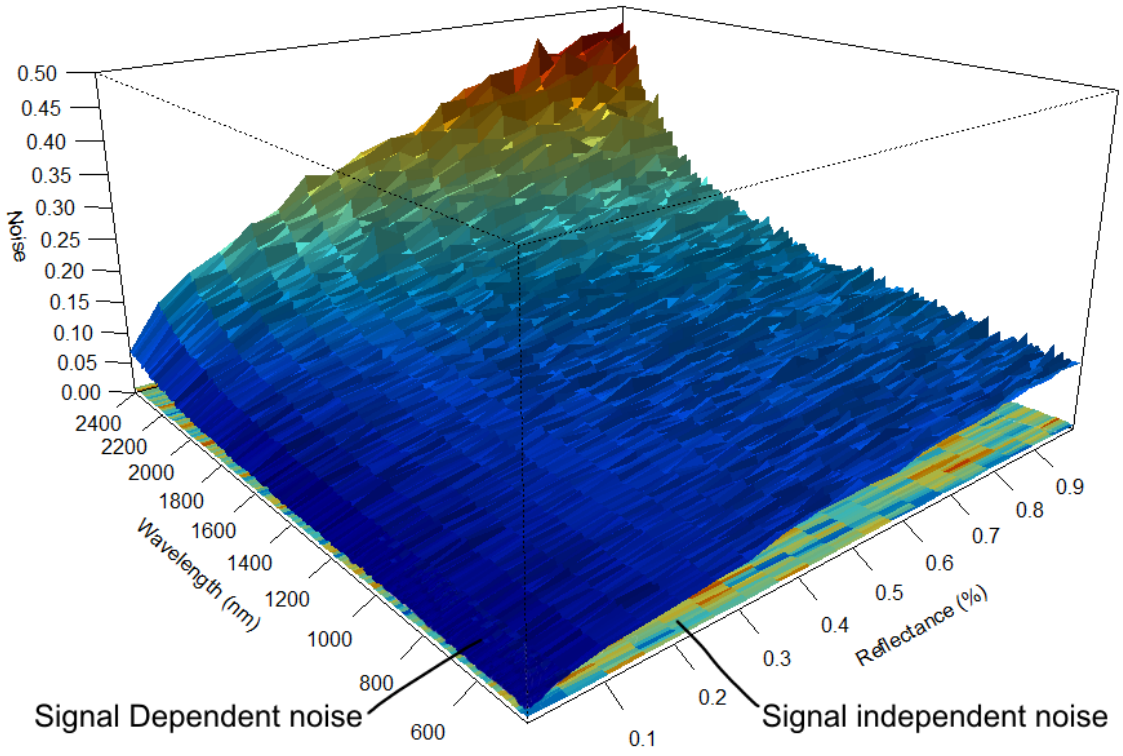

Figure 3.9: A 3-dimensional plot of the simulated noise. The upper surface is the signal dependent noise while the slightly obscured lower surface is the signal independent noise.

The script ends by printing to the screen the calculated SNR at the user defined band positions as a method of script validation. This calculated SNR is derived by dividing the mean signal value for the user specified SNR reflectance value (+/- 20%) and band position by the mean of the corresponding noise pixels (equation 3.5).

3.7 Summary

This chapter outlines the conceptual framework and rationale for the steps taken in the simulation of a hyperspectral satellite image. Six imaging spectrometer characteristics encompassing platform jitter/motion blur, spatial and spectral resolution, dual spectrometer wavelength coverage, and signal dependant and independent noise are presented. Open source program coding implements this framework to model a nominal sensor point spread function, spectral response function, spectral sampling intervals, and noise sources for both spectrometers of a dual system separately (see appendix A). For input, an airborne hyperspectral reflectance cube consisting of a single flight line or mosaic at oversampled spatially and spectrally resolutions is used to model the convolved coarser resolution and lower fidelity image. This input dataset must consist of a sufficiently broad spectral range beyond that of the desired spectral range of the output image. In the event that the input data are not broader the scripts will synthesize an image with abbreviated spectral range accordingly. The scripts are written to be flexible facilitating different user required characteristics to be defined.

The theoretical framework is intended to provide readers with a sense of the steps required to model data for remote sensing case studies and classification algorithm development. The outputs of the modelling create a synthetic image reflectance cube as well as signal dependent and independent noise images. Metadata including the band positions and spectral resolution are outputted in .CSV file format in the working directory.

The high dimensionality of hyperspectral data and the spectral and spatial oversampling necessary to simulate satellite imagery creates a significant challenge for computation. The use of the R-language for program coding is performance limited in this regard. Computation times obtained on a high performance personal computer using these scripts were measured in hours and days and not minutes.

Chapter 4: A Comparison of Mapping Arctic Lithologies using Simulated EnMap and Airborne Hyperspectral Imagery

4. 2 Introduction

Remote Predictive Mapping (RPM), the assimilation and interpretation of several forms of remote sensed sensor data alongside previously acquired field based observations, provides an objective analytical approach to regional analysis of rock properties (Harris, 2007), permits systematic generalization techniques to be employed (Smirnoff et al., 2012), and helps direct what regions of a study area require further investigation during fieldwork (Harris, 2008). RPM techniques output classified maps showing rock and/or topographic properties along with spatial representation of the uncertainties in the classification. Statistical analysis of these results provides a degree of control and measureable veracity. The ability to produce quality predictive maps prior to any fieldwork and aid in those areas that were not covered subsequently can thereby significantly reduce operational costs and produce more comprehensive maps.

The expected launches of several spaceborne hyperspectral imaging (HSI) systems will soon offer wide geographic coverage previously unavailable by existing HSI sensors. They also may provide enhanced capability to produce predictive mapping for geological campaigns. Given that these future sensors are anticipated to take an *open data policy* to geometrically-corrected surface reflectance data to non-commercial and commercial users alike (Guanter et al., 2015) they will play an important role in many applications. However these sensors have associated decreased spatial and spectral resolution and reduced Signal-to-Noise Ratios (SNR) compared to their airborne

counterparts that may make them unsuitable to specific applications and more appropriate to others. The question arises then what impact these performance parameters might have on the classification accuracy of geological predictive maps.

Currently planned for launch in 2019 is the German Aerospace Center's (DLR) so called high-resolution and wide-spectrum satellite mission: Environmental Mapping and Analysis Program (EnMAP) (Guanter et al., 2015). EnMap will be a push-broom HSI instrument with dual prism spectrometers, one for each of the VNIR and SWIR wavelengths ranges. It will collect radiance in the wavelength range from 420 to 2450 nm using 244 bands, 89 of which will in the VNIR (420 nm - 1000 nm) and 155 bands for the SWIR (900 nm - 2450 nm). The two spectrometers will have a spectral overlap of 100 nm. Spectral sampling distances will be 6.5 nm for the VNIR and 10 nm for the SWIR wavelengths. A 1024 spatial dimension detector array will produce a 30 km wide ground swath with a pixel size of 30 x 30 m. The SNRs are expected to be 500:1 (at 495 nm) and 150:1 (at 2200 nm) using a complementary metal oxide semiconductor (CMOS) and mercury cadmium telluride (HgCdTe) arrays for the VNIR and SWIR spectrometers respectively (Sang et al., 2008). EnMap will travel a sun-synchronous orbit at an altitude of 652 km.

This chapter examines the potential of this soon to be launched spaceborne HSI sensor and thereby the influence of lower SNR and resolutions on the discrimination of exposed lithological classes for predictive mapping. It compares and presents the results of two disparate classification approaches applied to simulated EnMap imagery to that derived using high resolution and SNR airborne HSI: ProSpecTIR–SPECIM (a proprietary

modified AISA sensor). A detailed 1:25,000 published geological map (Sherlock and Carpenter, 2003) was used to both train and validate the predicted maps.

The first classification approach involves a previously determined successful technique for when lithological unit rocks lie within mixed pixels. This process, outlined in Harris et al. (2006), uses a forward and subsequent inverse Minimum Noise Fraction (MNF) transform on the high order components to return the pixel spectra back to their normal spectral spaces but with reduced noise. This imagery is then unmixed using a matched filtering (MF) process and the resulting abundance images are thresholded to create a classified thematic map. The second approach uses known geological mapping band ratios and indices alongside a new trending non-parametric classifier showing promise: Random Forest (RF). Two methods are applied to assess which might be more successful at producing a representative predictive map when dealing with rock classes that contain high within class spectral variability at different resolutions.

4.3 Study area

The study area is located in the Canadian territory of Nunavut between the latitudes of 68°00’N and 68°10’N and the longitudes of 106°30’W and 106°40’W) (figure 4.1). The surveyed area covers roughly 20 km × 8 km in the Wolverine-Doris Corridor of the 80 km long Hope Bay Greenstone Belt: an area that lies directly north of the Arctic Circle and east of Bathurst Inlet. The belt is associated with the northern Slave Province of the Canadian Shield containing largely Archean granite-greenstone-metasedimentary terrane (Sherlock and Carpenter, 2003) and consists largely of mafic volcanic and intrusive rocks (Gibbins, 1987; Gebert, 1993) overlain in areas by a Holocene cover of clay, silt, minor

amounts of sand, rarely cobbles and boulders, and a few linear ridges of esker sediments (Kerr and Knight, 2001). The region contains shear-zone hosted vein-type orogenic gold (Au) deposits and potential mineralization within Volcanic Massive Sulphides (VMS) (Sherlock et al., 2003). Base metal mineralization formed at mafic/felsic contacts and seen as quartz carbonate veins (Gebert, 1993) led to the discovery of precious and base metal deposits and eventually two exploration mines: Doris and Madrid (Sherlock et al., 2012). Since the discovery of mineralization, the region has undergone systematic geological mapping within the last 50 years (Fraser, 1964; Gibbins, 1992; Kerr and Knight, 2001; Sherlock and Carpenter, 2003; Sherlock and Sandeman, 2004).

The landscape of the region is low undulating relief tundra with areas of small cliffs and ridges (Gibbins, 1987) and numerous disjointed lakes. Outcrop locations of varying shapes and sizes are distributed throughout upland terrain and interspersed with vegetation and the aforementioned glacial sediments. The area is north of the treeline and the limited vegetation consists of low sedges, cotton grass, and herbaceous and shrub plants.

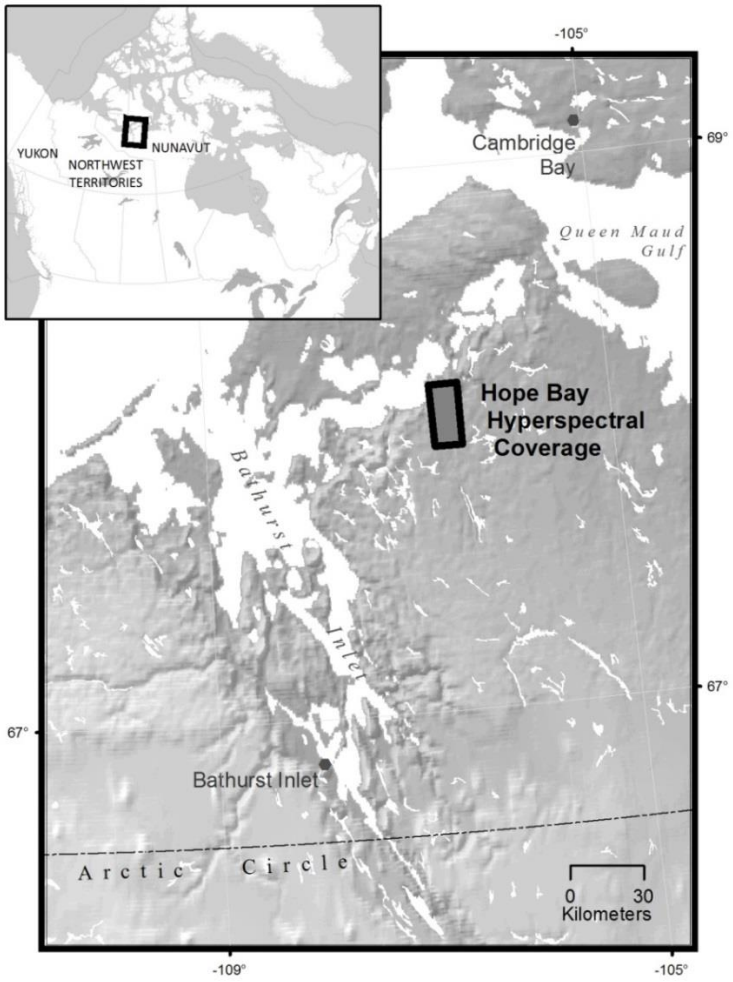

Figure 4.1: Location map showing the Hope Bay region in relationship to the rest of Canada (upper left) and the area covered by the ProSpecTIR–SPECIM hyperspectral survey used for EnMap simulation.

4.4 Methods

4.4.1 Data and preprocessing

The airborne data used in this study were acquired by a ProSpecTIR–SPECIM HSI system that consists of a dual co-aligned sensor assembly. The dual system consists of modified Specim AISA sensor designs: an AISA Hawk for the VNIR and an AISA Eagle

for the SWIR wavelengths. Both utilize push-broom designs that have a combined spectral coverage from 398 to 2446 nm. Radiance was collected in 355 bands of which 130 are VNIR (398-970) and 235 SWIR (970-2446). Both have selectable spectral resolutions: in this study they were collected at 4.4 nm for the VNIR and 6.3 nm for the SWIR. Acquired spatial resolution varies with aircraft altitude but was collected at 3 m for this region.

The imagery was collected by SpecTIR Inc. for the Geological Survey of Canada under the Strategic Investment in Northern Economic Development (SINED) program and supplied as an atmospheric corrected reflectance. Atmospheric correction was accomplished using the radiative transfer model: Moderate Resolution Atmospheric Radiance and Transmittance Model (MODTRAN). Because the surveyed area consisted of thirteen overlapping (minimum 30%) flight lines acquired on August 21 and the 23rd of 2009, it was necessary to combine them into a seamless mosaic of normalized relative radiometric values for complete coverage of the study region.

4.4.1 EnMap Simulation

The creation of a synthetic EnMap image was achieved through CRAN R scripts as detailed in the previous chapter (scripts are found in appendix B). These scripts model six sensor parameter considerations: sensor jitter, Point Spread Function (PSF), Spectral Response Function (SRF), dual spectrometer overlap, and additive signal dependent and independent noise. The workflow for the simulation is shown in the grey section of the complete methodology flowchart for this chapter (figure 4.2). The ProSpecTIR–SPECIM

airborne data described earlier were employed to model the EnMap data, as a spatial and spectral oversampled dataset is required to accurately synthetize the scene. Due to the complexities of modelling, no consideration was made for the increased atmospheric noise caused by an increased optical depth due to a satellite's relatively higher altitude.

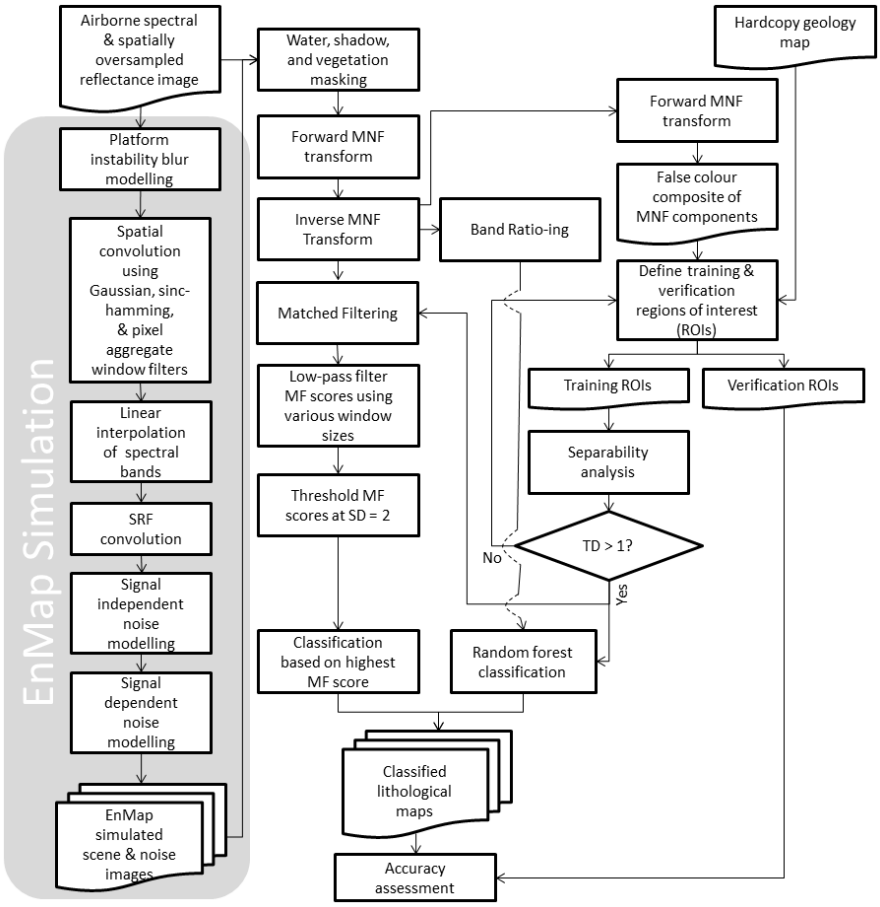

Figure 4.2: General processing workflow used for simulation of EnMap simulation through to lithological unit classification.

Studies simulating PSF responses typically make use one of two possible approaches: the first assumes that the measured ground within a pixel is equally weighted with no outside pixel influence (pixel aggregation of mean values). This approach is simplistic and as result is not computationally demanding. Because of the large volume of data and thereby

the required processing time required during simulation it may be the only sensible method. The second approach attempts to nominally simulate the PSF through convolving the input data using a Gaussian window function with the Full Width Half Maximum (FWHM) value equal to the spatial resolution of the sensor. Its shape is a muted form of the (ideal) response of a detector that because of the requirement for good frequency response follows a sync function (shown in figure 4.3). Unlike the sync function the Gaussian window terminates its ends at a 0 response. These zero response tails are necessary to produce a convolution window that can be applied digitally. Here EnMap scenes are simulated using these two methods: aggregation of the pixels and a Gaussian window, and include a third method to assess the influence of these differing methods might have on classification results. The third approach introduced attempts to simulate one lobe of the spatial response associated with the ideal sinc function and includes a zero response termination at its edges (see the grey line in figure 4.3). This is achieved through the multiplication of the hamming (equation 3.3) and sinc window functions (equations 3.1) (Smith, 1999). This resulting sinc-hamming window function closely resembles the sinc function but without the outward infinitesimal response. All three modelled PSF are shown in figure 4.3 with the ideal sync function as a reference. All three modelling approaches are utilized in this study to understand the visual and statistical difference they have on classification outcomes.

Prior to the spatial convolution, a Gaussian shaped moving 3x3 windowed kernel filter was applied to the input imagery to simulate the effect of high frequency random movements caused by onboard electronics, typical for spaceborne platform instability (Holst, 1998).

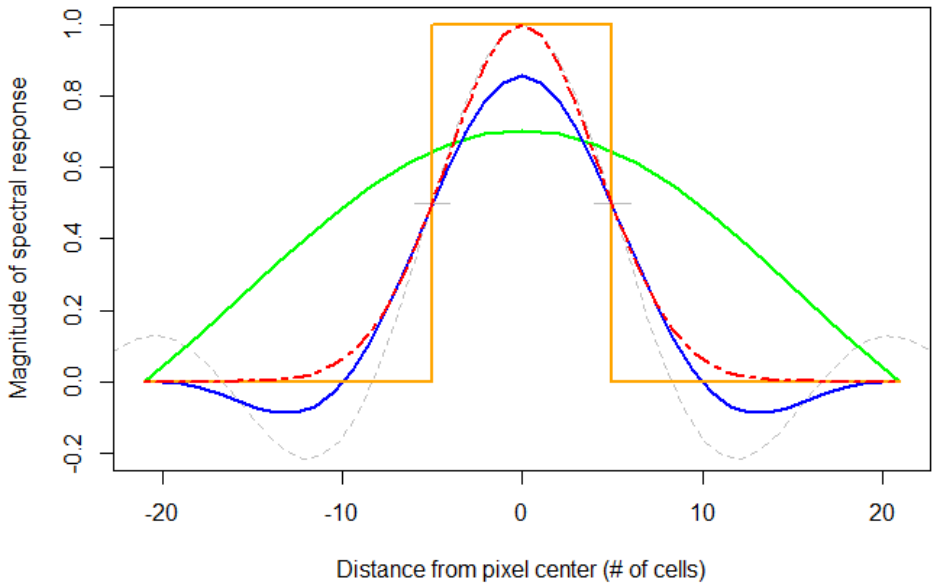

Figure 4.3: Three modelled PSFs with the ideal sync function (in grey) for reference. The Gaussian is shown in red, the sinc-hamming in blue, and the pixel aggregate in orange. Although not used as a modelled PSF, the hamming window filter is shown in green to demonstrate how it appears before multiplying it with the sinc window function.

4.4.2 Data Processing - Image Preparation

Masking

Both the mosaicked airborne and simulated EnMap scene underwent a series of common pre-processing steps. The first removed all non-geological related content through masking to help improve classification accuracy. This included a water mask determined by thresholding band values associated with a near infrared water absorption feature that occurs at 1125.5 nm for the airborne data and 1130 nm for the EnMap data. All pixels in which the reflectance was lower than 9% were determined to represent areas of water for both datacubes. This value was determined through iterative changes of values and subsequent inspection of the imagery and histograms. This water mask, given its low

reflectance, was found to also act as a very conservative shadow mask. No further masking of shadows was applied as Harris (2005) found coarse grained rocks with corresponding low reflectance, as is the case with the Proterzoic diabase rocks in this study area, may also get masked if attempts are made to mask shadows.

Rock outcrop areas were further isolated by removing pixels dominated by vegetation using a Normalized Difference Vegetation Index (NDVI) (equation 4.4). NDVI takes advantage of a marked spike in spectral reflectance between the red and near-infrared wavelengths associated with the spectra of vegetation. Index values range from -1 to 1 where heavily vegetated pixels are attributed with higher positives values and vice versa. For the EnMap simulated images the red band was specified at 680 nm (Band 41) and the near infrared band values at 745 nm (band 51). For the ProSpecTIR–SPECIM airborne data similar values of 681.5 (band 62) and 748.5 nm (band 76) were used. Pixels containing little or no vegetation cover were separated from vegetated pixels by thresholding NDVI values at 0.478 for both the EnMap simulated scene and airborne imagery. This threshold value was again determined by means of visual inspection of the imagery and corresponding histograms.

$$NDVI = \frac{R_{NIR} - R_{red}}{R_{NIR} + R_{red}} \qquad \textit{Equation 4.4}$$

where *R* are reflectance values of specified near infrared (*NIR*) and red (*red*) bands.

Areas left unmasked after these masking procedures corresponded well with areas marked as outcrop on the geological map with the exception of areas visibly dominated by glacial overburden and dry, non-photosynthetically-active vegetation. Attempts at

masking these glacial overburden areas were unsuccessful as they were intermixed with and spectrally similar to the bedrock areas.

Band Removal

The following preprocessing step involved the removal of undesired bands. For the EnMap imagery this included the removal of 10 SWIR bands within the overlapping region of the two spectrometers (900-1000 nm). Additional bands were removed from both the ProSpecTIR–SPECIM airborne and simulated EnMap datacubes as a result of the absorption of water vapor and major gases that result in higher band noise (van der Meer, 2009). These included bands within the spectral ranges between 1325-1488 nm and 1780-1970 nm.

Noise Reduction (Inverse MNF)

Noise is unfortunately an inherent disadvantage of hyperspectral imagery as it is considerably higher than multispectral imagery. This is owing to the system compromises that are made to achieve a narrow band width. Noise comes in two forms: random and fixed-pattern (Acito, 2011). Random noise can be considered the irregular white noise whereas fixed pattern noise is produced by constant detector errors. Fixed pattern noise is often measureable and removed through systematic and radiometric correction (and often supplied as ‘Level 1’ satellite data). However the random noise is the more dominant of the two in modern HSIs and difficult to remove given its unpredictable behaviour (Acito, 2011; Liu et al., 2014). Noise impairs interpretation and thus image enhancement seeks to alter values in an attempt to diminish its effect.

A forward followed by inverse Minimum Noise Fraction (MNF) transform (Green et al., 1988a, 1988b; Harris, 2006) was applied to the EnMap and ProSpecTIR–SPECIM airborne imagery to reduce the random noise and improve the SNR. The forward MNF is a two-step data reduction and noise transformation process. The first step decorrelates and rescales the noise in the data based on an estimated noise covariance matrix. The second step applies a Principal Component (PC) transformation to this noise-whitened data. This second step orders the output MNF components decreasingly by image quality, where the lowest components represent the noise within the imagery. The number of components is equivalent to the number of input bands. In this study, a forward MNF transform was applied to the unmasked areas and high ordered components were visually inspected. Any high order components that were found to still be noisy were low-pass filtered using 3x3 window. The remaining MNF components containing mostly noise (low ordered) were identified using a plotted scree test (figure 4.4) and then discarded. These represented all but the top 50 components of the 194 bands for the EnMap imagery and 50 of the 301 components of the ProSpecTIR–SPECIM imagery. An inverse MNF transformation restored the signals to the original spectral signatures but now utilizing only the remaining significant components and thereby removing noise from the original imagery.

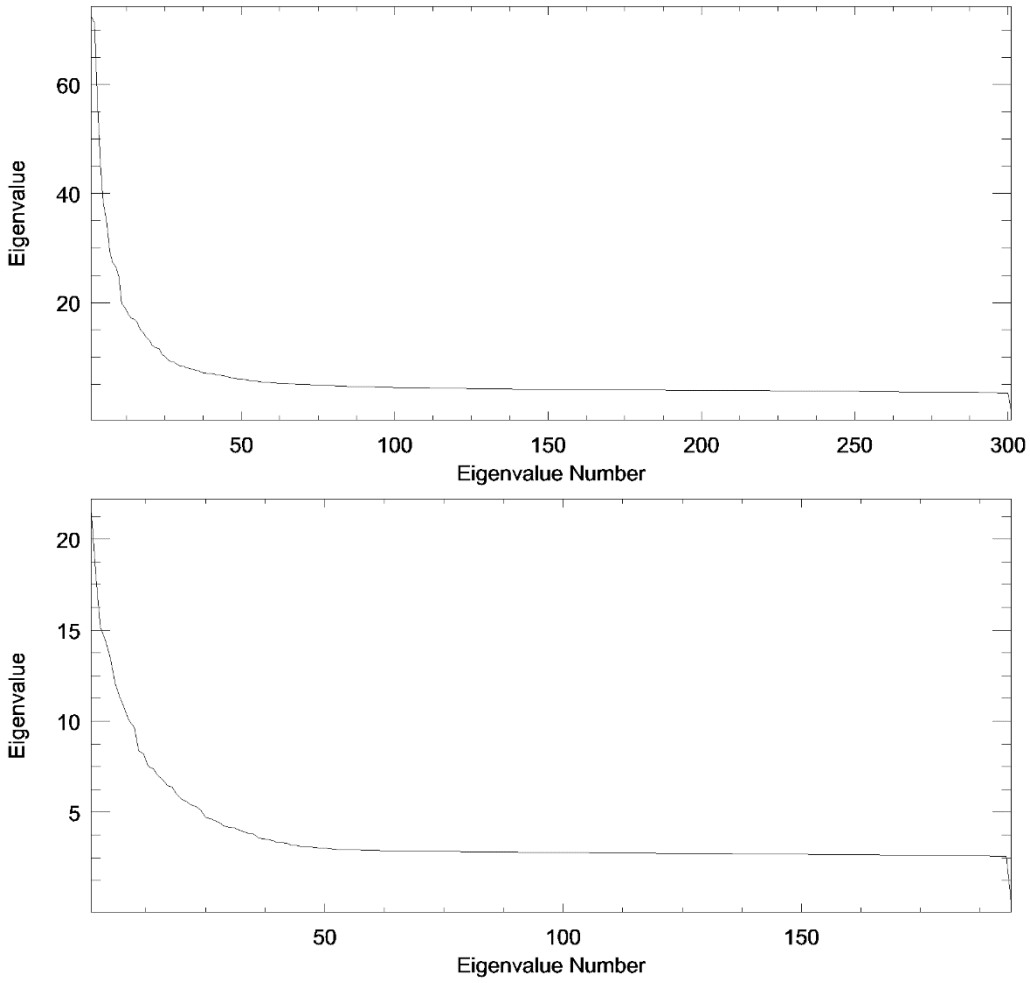

Figure 4.4: MNF Eigenvalue plots for the Wolverine-Doris Corridor AISA ProSpecTIR-SPECIM (top) and EnMap (bottom) imagery. For both sensors the top 50 Eigen values (components) were used in the inverse MNF transform.

Band Ratios

The high dimensionality (bands) of hyperspectral data poses a problem with regards to pixel based supervised classification accuracy. With an increase in dimensions comes a requirement to increase the size of the training data to maintain classification accuracy, a situation known as the Hughes Phenomenon or the Curse of Dimensionality (Richards & Richards, 1999). Since increasing the size of the training areas sufficiently is likely not

possible given the requirement for more legacy data or continued fieldwork to ground truth the data. To solve this problem data reduction strategies are frequently applied. One such approach, and common to geological studies, is the application of band ratios and indices. These seek to normalize the intensities among different bands and thereby help reduce illumination effects associated with topographic slope and shadows that increase spectral variance. Targeted geological minerals are also discriminated by placing band(s) with known absorption properties as the numerator and bands known to have high reflectance properties as the denominator. The effect is to produce a single image with digital numbers (DN) for each desired mineral. Those DNs with high values indicate the likely presence of the mineral and vice versa. Band indices similarly achieve the same result except that they incorporate band math that utilizes more than two bands as ratios do.

Fifteen common geologic ratios and indices were used on both the EnMap and ProSpecTIR–SPECIM imagery (Harris et al., 1998; Rowan & Mars, 2003; Kalinowski & Oliver, 2004) and stacked as a single multi band image for preparation for subsequent classification. These ratios and the mineralization they are intended to map are shown in table 4.1.

Table 4.1: List of common ASTER and Landsat band ratios for targeted mineralization spectral features. Included are the wavelengths for which they were created for and the corresponding bands used in this study.

Ratio	Multispectral Bands	Center wavelength (μm)	EnMap Bands Used (μm)	ProSpecTir Bands Used (μm)
Ferric Iron	2/1	**0.66/.56**	0.667/0.563	**0.6577/0.5578**
Ferrous Iron	5/3 + 1/2	**2.165/0.82 + 0.56/0.66**	2.160/0.823 + 0.563/0.667	**2.16.5/0.8217 + 0.5578/0.6577**
Gossan	4/2	**1.65/0.66**	1.650/0.667	**1645/0.6577**
Ferric Oxides	4/3	**1.65/0.82**	1.650/0.823	**1.645/0.8217**
Carbonate Chlorite	(7+9)/8	**2.26+2.395/2.365**	2.260+2.390/2.365	**2.2584+2.3961/2.3648**
Epidote	(6+9)/(7+8)	**2.205+2.395/2.26+ 2.365**	2.205+2.390/2.260+2. 365	**2.201+2.3961/2.2584+ 2.3648**
Amphibole (MgOH)	(6+9)/(8)	**2.205+2.395/2.365**	2.205+2.390/2.365	**2.2021+2.3961/2.3648**
Dolomite	(6+8)/7	**2.205+2.365/2.26**	2.205+2.365/2.260	**2.2021+2.364.8/2258.4**
Sericite	(5+7)/6	**2.165+2.26/2.205**	2.160+2.260/2.205	**2.164.5+2.2584/2.2021**
Alunite	(4+6)/5	**1.65+2.205/2.165**	1.650+2205/2.160	**1.645+2.202.1/2.1645**
Phengitic	5/6	**1.65/2.205**	1.650/2.205	**1.645/2.2021**
Muscovite	7/6	**2.26/2.205**	2.260/2.205	**2.258.4/2.2021**
Kaolinite	7/5	**2.26/2.165**	2.260/2.160	**2.258.4/2.1645**
Ferric Iron (Landsat)	3/1	**0.66/0.485**	0.667/0.485	**0657.7/0.483.9**
Clay (Landsat)	5/7	**1.65/2.220**	1.650/2.220	**1.645/2.2208**

Reference data preparation

The classification of imagery requires defined training areas of representative rocks for each lithological class to create a thematic map. The statistical assessment of the output classification requires verification areas. Ideally ground truth training and verification should be defined through fieldwork however obtaining this information requires expensive fieldwork and was out of scope of this study. Instead these locations were determined by referencing a scanned hardcopy published bedrock geology map (Sherlock

and Carpenter, 2003) and matching locations easily delineated as homogenous outcrop in the hyperspectral imagery. This was accomplished using a false colour composite of the highest three components of the forward MNF ternary images discussed earlier. These components represent the most variance from the original higher dimension data and helped to distinguish outcrops from the glacial sediments and soil.

The same spatial extents of the training and verification sites were used for the classification of both the ProSpecTIR–SPECIM and EnMap data. Identified homogenous areas were randomly separated into training (60%: n = 822 and n = 90,306) and validation pixels (40%: n= 527 and n = 61,332).

Class designations of the sites were determined with reference to the published map. This map was unlike most geological maps, which typically infer units over areas of glacial sediments, overburden, and vegetation, as it contained detailed delineation of the outcrop locations from the surrounding topography. The scanning and folding of the map caused distortions in the map image and thereby had to be georeferenced using a third-order polynomial transformation with 92 locations that were easily identifiable in both a true colour ProSpecTIR–SPECIM image and geological map. These ground control points (GCPs) had a Root Mean Square (RMS) error of 1.8 m sufficient to ensure matching of the high (3 m) spatial resolution airborne data. This map covered the entire surveyed area. It also showed favourable correlation between the locations of mapped bedrock outcrop and those identifiable in the imagery.

Six lithological classes were generalized from the 12 classes found in the geological map. This reduction in class numbers is given by their identified more broad lithological

characteristics. For example two felsic metavolcanics units are mapped and designated as "unit 4" but are subdivided by their clastic properties (coherent and clastic). In all map units with the exception of Proterozoic diabase, the geologists have indicated that each unit consists of further subdivision based of rock compositions with letters. The six classes in order of their spatial dominance are: mafic metavolcanic (map unit 1), Proterozoic diabase (map unit 11), felsic metavolcanic (map unit 4), late, post-volcanic intrusions (map unit 10), early, syn-volcanic intrusions (map unit 8), and early syn-volcanic sedimentary rocks (map unit 5) (table 4.2).

Table 4.2: Lithological unit classes and matching training and verification sizes (number of polygons/number of pixels) for both datasets.

	EnMap	ProSPecTIR	EnMap	ProSPecTIR
Lithological Class	**Training**	**Training**	**Verification**	**Verification**
Mafic metavolcanic (Mmv)	88/435	88/42,959	67/249	67/28,451
Proterozoic diabase (Pd)	23/78	23/9,925	21/73	21/8,884
Felsic metavolcanic (Fmv)	24/99	24/12,660	25/81	25/10,621
Late, post-volcanic intrusions (Lpvi)	18/105	18/11,771	16/64	16/6,435
Early, syn-volcanic intrusions (Esvi)	18/62	18/7,381	15/44	15/4,527
Early, syn-volcanic sedimentary (Esvs)	18/43	18/5,610	16/16	16/2,414

Significant effort was made to ensure that there was large sampling to sufficiently support the inherent spectral variability for each class given the noted lithological class rock variability, weathering, and differing illumination conditions caused by diffuse illumination and/or adjacency effects.

4.4.3 Supervised Classification of Lithological Units

Matched filtering classification for Lithologic map units

Matched Filtering (MF) (Boardman et al., 1995) was used to classify the imagery using averaged spectral signatures from the training areas characterizing each lithological unit. MF uses these known spectral signatures for each class to decompose pixels and derive images, termed as either abundance, fraction(al), score, or probability maps, to indicate the amount of contribution or match for each surface material in each pixel. The total number of derived abundance images created for classification is equivalent to the number of classes being mapped. Unmixing pixels this way is particularly advantageous for mapping with coarser resolution data because it can resolve mixtures of materials within these pixels by detecting and matching diagnostic spectral features of the targeted classes and it does not require knowledge of all the endmembers within the image (Mustard and Sunshine, 1999; Keshava and Mustard, 2002). It attempts to ignore unknown spectra and detect the rare spectra of the scene (Shippert, 2003). In areas of lithologies with complex compositions of various minerals and widespread lichen, common within the Canadian arctic, MF may be an optimal algorithm as it has shown robust results in identifying targeted mineral spectra (Harris et al., 2005, 2007 & 2010).

Specifying a threshold value within the abundance images is required to separate lower value matches, indicating less certainty, from pixels with higher values and thereby producing a binary image for each class. Thresholding and removal of lower abundance pixels was required to isolate pixels from the overburden and dry, non-photosynthetically-active vegetation. Subsequent to each class derived abundance image

having a threshold applied, pixels that were assigned to more than one class were then allocated to the class with the highest abundance value to produce a multiclass thematic map. Thresholding and assigning pixels to their designed class was accomplished through CRAN R scripting (see Appendix C).

Random Forest classification for Lithologic map units

Supervised classification was also conducted using the machine learning algorithm Random Forest (RF) (Beiman, 2001; Breiman & Cutler, 2016) using the fifteen band ratios and indices in table 4.1. RF is a classifier showing promise for geological mapping particularly if the training data are non-normally distributed (Cracknell and Reading, 2009; Harris et al., 2014; Waske et al., 2014; He and Harris. 2015). It uses several run decision tree models (typically in the hundreds and hence the term "forest") to produce a new more accurate classification based on the majority of pixel votes. During each of the decision tree classification models, a specified number of randomly selected bands are used. The training data are also randomly split with replacement into training and verification data (termed bootstrapping). The out-of-bag verification data (those locations not used in training the classification) are used to produce an unbiased accuracy assessment and thus RF does not need a separate verification dataset once the classification is complete (Beiman, 2001).

Although RF is insensitive to outliers in the training data, it does provide the advantageous ability to identify these areas after classification to highlight either poorly represented or mislabelled training data. These locations can be corrected, removed, or

replaced to enable the classification to be rerun for greater accuracy (Breiman & Cutler, 2016).

The importance of each band, in our case each ratio or index, in predicting an accurate classified map can also be calculated using RF. By re-running the same classification with reordering of a single band's values and then assessing the change in accuracy, RF can, once the same is repeated for all bands, establish the relative importance of that particular band. When this substitution of values produces a significant drop in classification accuracy it indicates and increases its importance and vice versa for low relevance bands (Horning, 2010; Breiman & Cutler, 2016). This rate of classification error is recorded as a mean decrease in accuracy for each band.

Band importance in the RF classification is also supplied through a 'Gini importance' value. This variable importance metric is calculated by adding the weighted impurity decreases, measured by comparing the difference in classification accuracy between the parent and subsequent splits, for all nodes averaged over all trees in the forest. Although the Gini index is a commonly used metric as it often orders variables the same as the mean decrease in accuracy it could bias some predictive variables (Strobl et al., 2007).

RF was implemented in this study using a modified CRAN R script written by Horning (2013). Two thirds of the training data were used to build the RF model and the leftover third of the sample data were used for estimating the error of each decision tree (out-of-bag samples). The number of trees grown in the forest was set to 300 and an assessment was made to check that this was sufficient by confirming that there was indeed a stabilized error rate using an x/y plot of error % to number of trees. The number of

variables (ratio/indices bands) for each tree of the forest was determined using the common application of the square root of the input variables, in this case three. Although RF is said to provide an unbiased cross-validation error the same verification data used earlier for the MF classification was employed to conduct a confusion matrix. This was accomplished to ensure both classification approaches were equally assessed for comparison. Similarly the same method of thresholding class probability maps used in the MF classification was applied to separate correctly classified pixels from the overburden.

4.5 Results

4.5.1 Training Area Spectral Signatures

The mean reflectance spectra of training samples extracted from the hyperspectral imagery (inverse MNF datacubes) for the six lithological classes are shown in figure 4.5. Between the two imaging spectrometers these signatures align very closely, the exception being that as the spectral and spatial resolution decrease there are varied (across the spectral range) altered reflectance intensities for each of the classes and a loss of small fluctuations (being either noise or genuine absorption features) with the lower resolution EnMap (Gaussian PSF) data. The change in spectral intensities is likely owing to pixel mixing with neighbouring landscape surfaces (Staenz et al., 2001) however the change varies greatly depending on the class.

In general however, the spectral signatures of classes, within each imagery set, appear similar to one another. The largest difference is their distinct (broad) spectral differences

in intensities owing to the light or dark properties of the rocks. The largest of these contrasts occur at 440-800 nm, 960-1150 nm, and in the SWIR at 2020-2045 nm. Minute spectral features are absent and likely caused by the averaging of training pixels for classes. All classes exhibit a characteristic slope between 670 and 740 nm likely owing to inter-pixel mixing of vegetation despite efforts to mask it. Beyond these noticeable general attributes there are a few unique class dependent spectral qualities.

The Late post volcanic intrusion (in blue in figure 4.5) contains subtle relatively deeper absorption features at 1050, 1250, and 2020 nm and a comparatively steeper slope between 470 to 560 nm (indicating possible iron oxides). The Proterozoic diabase (in red) contains the lowest spectral intensity throughout the wavelength range and minor absorption features at 450, 700, and 1200 nm. The early syn-volcanic sedimentary rocks (purple line) consist of argillite and greywacke (Sherlock and Carpenter, 2003) and despite their fine grained nature these dark coloured rocks have decreased spectral reflectance such that only at 2200 nm there is a relative absorption feature (indicating possible mica content) and the weakest slope between 470-560 nm. Early syn-volcanic intrusions (green line) are characterized by a broad spectral absorption centered at 1170 nm and a lack of an absorption feature at 1000 nm. The mafic metavolcanics (cyan line) dominate the study area and are visibly the lightest coloured rocks (and slightly more light orange). They exhibit a spectral peak at 610 nm with comparatively weaker rise from 680 to a lower NIR reflectance, (possibly due to a lack of any vegetation on the outcrops). Felsic metavolcanics (in indigo) show a nearly identical signature as the early syn-volcanic sedimentary rocks (similar in intensity as well) and no doubt added to classification confusion between these classes.

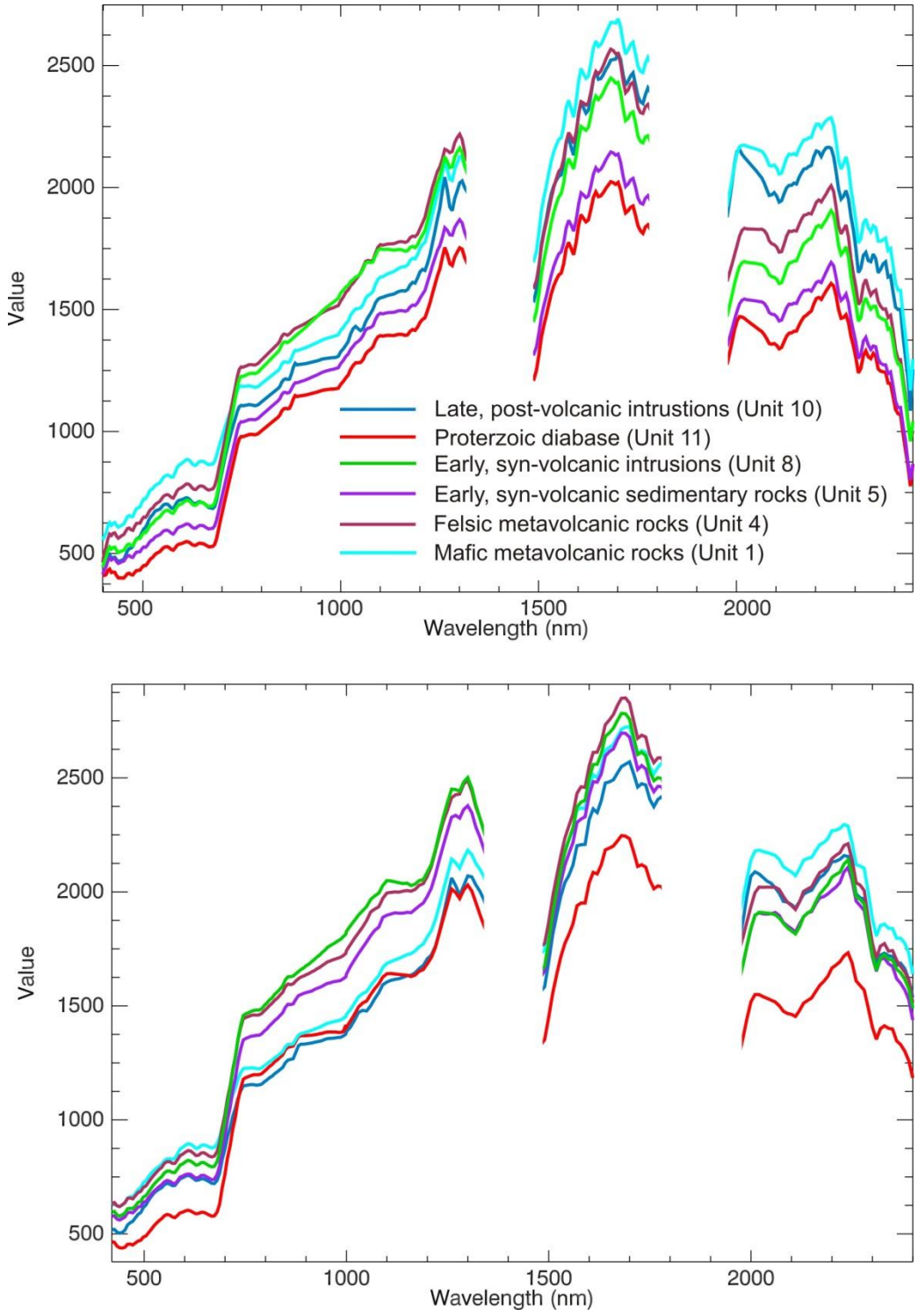

Figure 4.5: Averaged spectral signatures obtained from the training areas for each lithological class extracted from the ProSpecTIR–SPECIM (top) and the simulated EnMap (bottom). Atmospheric absorption bands have been removed.

4.5.2 Evaluation of Training Area Separability

Transformed Divergence (TD) statistics (Swain & King, 1973; Richards & Richards, 1999) were used to compute class separability between mean spectral signatures pairs and to confirm any relationship there might be with the choice of classification approach. TD gives a measure of probability of correct classification. Values range between 0 and 2. A TD value >1.9 indicates excellent classification accuracy, values between 1.5 and 1.9 moderate-to-good separability and less than values < 1 indicate poor separability and suggest consolidating classes (Jensen, 1987).

Figure 4.6 shows box and whisker plots of TD values of the band ratios and indices for both the ProSpecTIR–SPECIM and simulated EnMap imagery. Evident are four characteristics: TD values in all cases are moderate to excellent, there are significantly higher TD separability values within the EnMap data for all classes, all are near or at 2.0, and there is a broader range of TD values with the ProSpecTIR–SPECIM data. Also notable is that there is not a shared relationship of lower TD values for classes between the high and low resolution imagery. The lowest TD value in the EnMap data, between late, post-volcanic intrusions and mafic metavolcanic rocks (TD = 1.80), does not share a corresponding relative low TD value compared to the other classes in the airborne data. In fact this unit has the highest TD values of the classes for the ProSpecTIR–SPECIM imagery.

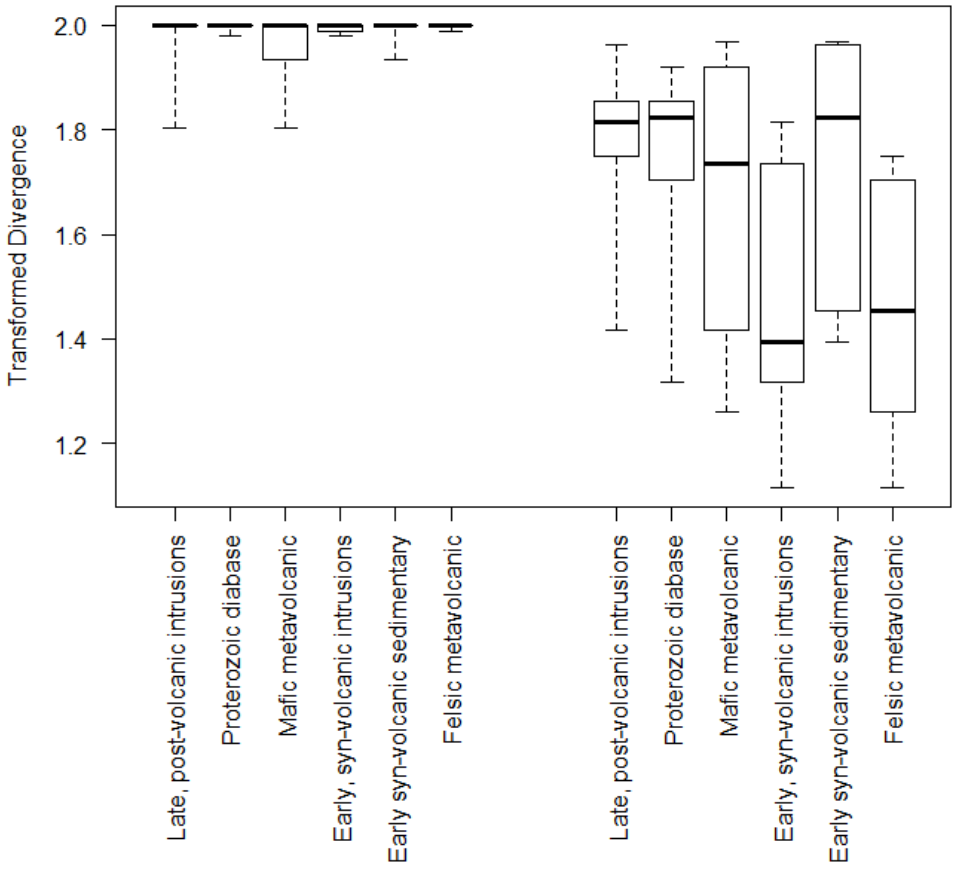

Figure 4.6: Box and whisker plots of Transformed Divergence (TD) values for all lithological classes using all band ratios and indices for (a) simulated EnMap (Gaussian PSF) on the left and (b) ProSpecTIR–SPECIM data on the right. Plotted are the mean value as the thick central line, min and max values as whiskers, and quartiles as the box.

Classification accuracy is also affected by spectral variability within the training data for each of the defined classes. With increased variability comes lower classification performance. In some cases the spectral variability may necessitate including separate representative training classes for a single geological unit to achieve more accurate results especially if the training data has a multimodal distribution (Behnia et al., 2012; Stevens et al., 2013; Harris et., 2014). Within class spectral variability was measured by calculating the mean of standard deviations (SD) for all inverse MNF bands and for the

ratios and indices as well. These SD values for both sensors for the band indices are shown in figure 4.7 and the inverse MNF bands in figure 4.8.

Most apparent in the SD of classes is the notable decrease in values between EnMap and the ProSpecTIR–SPECIM data for the band indices data. A weaker relationship of decreased SD with the EnMap data also exists for the inverse MNF and indices.

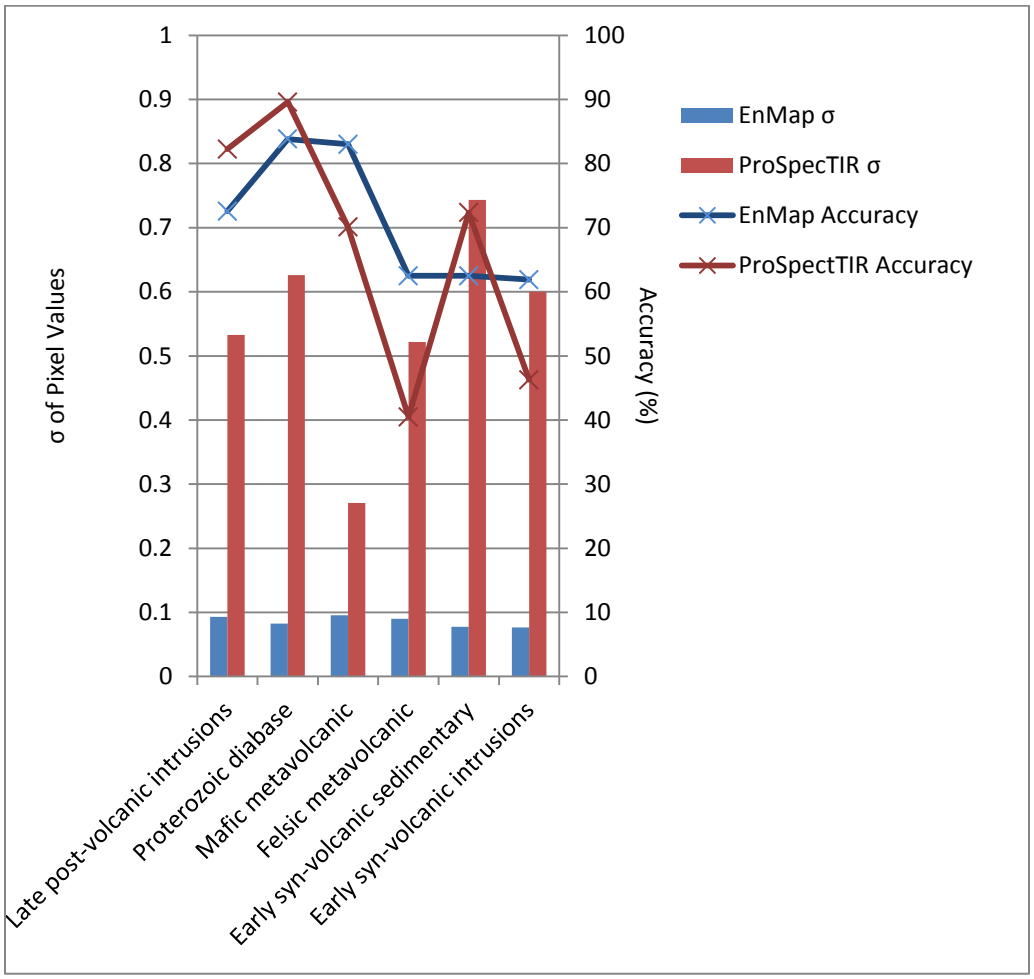

Figure 4.7: Standard deviation values found in each class training dataset for the band ratios and indices for both the EnMap (Gaussian PSF) and ProSpecTIR–SPECIM data (boxes). Also shown are the class accuracies for the RF classification using this data (lines).

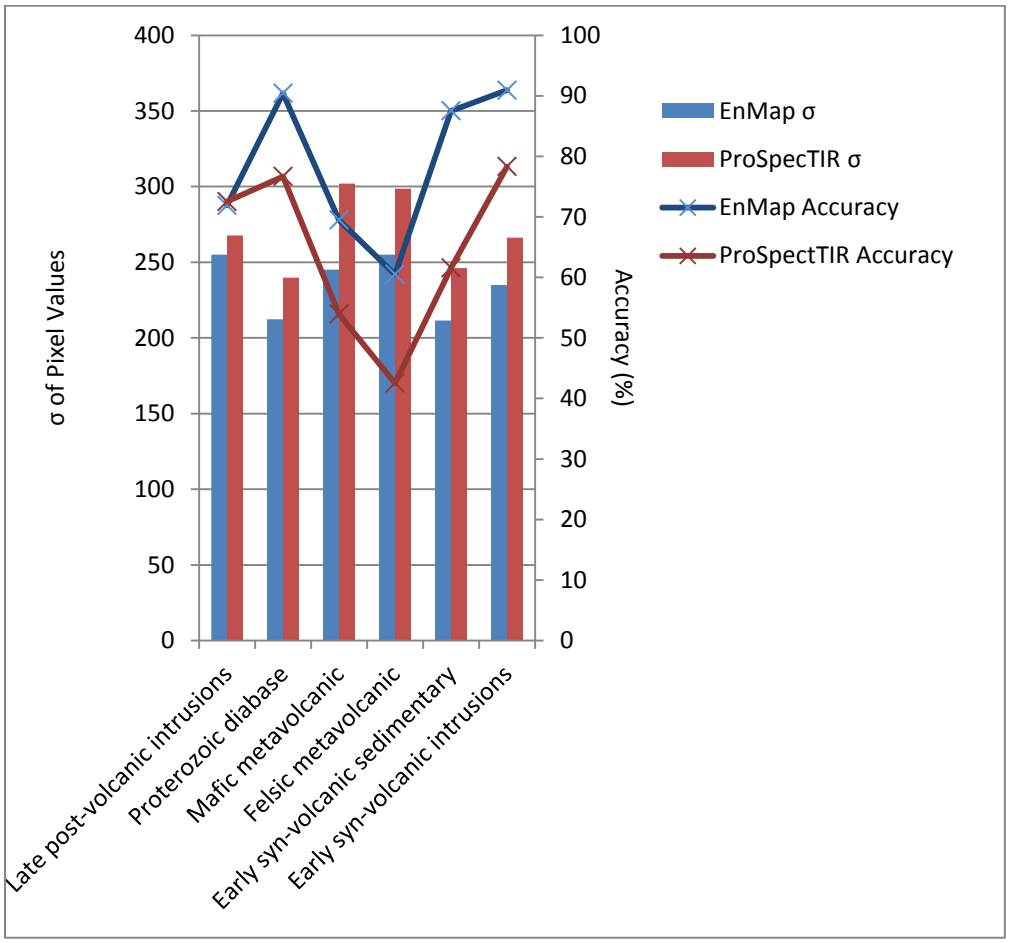

Figure 4.8: Standard deviation values found in each class training dataset for the inverse MNF imagery for both the EnMap (Gaussian PSF) and ProSpecTIR–SPECIM data (boxes). Also shown are the class accuracies for the MF classification using this data (lines).

Another evident difference between the sensors includes the mafic metavolcanics, where in the EnMap data the SD values are relatively higher compared to the other classes however in the ProSpecTIR–SPECIM data they are lower. It appears with this unit there is already a lower amount of mineral (spectral) variability within the airborne data that is not reduced with decreased resolution.

As expected, the results of the TD analysis described in the above section indicate there is a strong inverse relationship with these values and SD values: where there are high SD

values there are lower TD values and vice versa. For example, the mafic metavolcanic class shows high SD and low TD values in both sensor datasets for the band indices whereas Proterozoic diabase and early syn-volcanic intrusions had lower SD with corresponding higher TD values. The differences between classes appear to be consistent for the two sensor datasets.

4.5.3 Matched Filtering Classification

Matched filtering produced abundance images for each of the lithological classes. As mentioned elsewhere, after MF was performed the resulting scores required thresholding to refine the results to those pixels with a higher degree of certainty. The second standard deviation (~2.2% of the values) of each MF image (Harris et al., 2005; 2006, 2010) was determined to be the optimal value found through iterative visual and histogram analysis. Visually this was achieved by ascertaining the value separated exposed bedrock pixels from those that were easily identifiable such as glacial overburden.

Matched filtering supervised classification for the EnMap and ProSpecTIR–SPECIM data (figure 4.9) were compared with the verification areas as confusion (error) matrices (tables 4.1 and 4.2). Reported overall classification accuracies for the EnMap (Gaussian PSF) and ProSpecTIR–SPECIM data were 82.2% and 67.2% respectively. Kappa coefficients (k), a measure of comparing the classification to a random classification while considering not just overall accuracy but map unit correlation as well (Congalton, 1991), were 0.76 for EnMap and 0.58 for the ProSpecTIR–SPECIM results. According to Landis and Koch's (1971), interpretation of the kappa coefficient values ($0 < 0.00$ is

poor, 0.00-0.20 is slight, 0.21-0.40 is fair, 0.41-0.60, is moderate, 0.61-0.80 is substantial, and 0.81-1.00 is almost perfect) the EnMap classification is *substantial* whereas the ProSpecTIR–SPECIM classification is *moderate*.

For the EnMap predictive mapping the class accuracy ranged from 73% for the early syn-volcanic sedimentary rocks to 100% for the Proterozoic diabase. For the airborne modelled results the class accuracy for the individual classes ranged from 53.1% for the felsic metavolcanics to 94% for the Proterozoic diabase. In both cases the highest class accuracy was achieved with the Proterozoic diabase rocks which is not surprising since they had the most distinct spectral features, are also characterized by a dark gray to black colour, and have an overall lower reflectance throughout their signature. Their dark (low albedo) visual characteristic likely allowed the geologists to delineate them through air photograph interpretation and thereby with greater confidence in the field. As a result there is less expert "geologic interpretation" required and thereby good agreement in the classification of the hyperspectral imagery with this unit.

As the case with the EnMap the lowest classification accuracies occurred with the early syn-volcanic sedimentary rocks. The spectral signature of this unit is the most similar to the other class spectra. Additionally, there was a high degree of spectral similarity with this unit and the overburden (no class used during classification) as the overburden areas were often classified as this unit. This was more pronounced prior to the early syn-volcanic sedimentary rocks class' probability images being thresholding. This is likely owing to the high clay content within the overburden. Further efforts, beyond this study,

could perhaps include an overburden/dry, non-photosynthetically-active vegetation class to help distinguish these areas.

Confusion matrices and TD values show there is a high degree of confusion between this early syn-volcanic sedimentary rock class and the felsic metavolcanic rocks (see tables 4.3 through 4.7). This is most likely caused by the limited available number of training areas for the early syn-volcanic sedimentary rock (18 polygons and 43 pixels for EnMap and 18 polygons and 5,610 pixels for ProSpecTIR–SPECIM). Given that supervised classification results are heavily dependent on the quality and number of training areas this problem is more pronounced with the EnMap imagery. Despite the ProSpecTIR–SPECIM and simulated EnMap data containing the same geographic extents for their training data, the coarser resolution further limited the number of available training pixels. This class has a proportionately higher error of commission or in other words, a high rate of false positives for this unit. Overestimating rare materials in hyperspectral scenes is an identified weakness of using MF (Shippert, 2003).

Much of the incorrectly identified pixels, especially those for the ProSpecTIR–SPECIM imagery, can be attributed to spectral difference across flight lines that make up the multiple flight line mosaic. Since the mosaic includes imagery collected over three separate days and times (see appendix A) illumination conditions changed such that the spectral signatures for the eastern quarter of the mapped region overestimate the mafic metavolcanics (map unit 1), and under estimate the late, post-volcanic intrusions (map unit 10). Fortunately, the flight line north to south orientation aligns well with the dominant geological trend in the mapped units and this problem is minimized.

Spatial Filtering of Matched Filtering Abundance Images

Visual inspection of the output classification maps showed that the ProSpecTIR–SPECIM data had a high degree of isolated classified pixels that did not resemble actual geologic differences and appeared to be more related to classification noise. This noise may indeed be correctly classified pixels that are intermittent small locations of rocks within a larger prevailing geologic unit. They may be located at their corresponding locations through a geological transition or processes (as is the case with clastic facies of felsic meta-volcanics) and chosen not to be mapped by the geologist for thematic representation. Spatial filtering of the matched filtering abundances using progressively increased kernel sizes removed these pixels and improved the spatial cohesiveness of the classification (Harris et al., 2005; Brown et al., 2007). The subsequent reported classification accuracy improved from this treatment (from 67.2% (k: 0.58) to 71.4% (k: 0.63)). Once plotted the kernel size and overall accuracy showed a logarithmic relationship: there was an increase in accuracy with an increased filter window but with a progressive but slowing rate of improvement peaking at an increase of 4.2% using a 17 x 17 kernel size (see figure 4.11). Spatial filtering of the EnMap MF abundances images varied the overall accuracy depending on the PSF function considered. The Gaussian and pixel aggregation PSF results both decreased with filtering however the Sinc-Hamming showed a minor improvement (3.85%) that peaked early with a 3 x 3 kernel followed by a drop with any increment in size (see figure 4.11). In all cases, filtering of the scores reduced the SD of the MF values.

The maps with the highest overall accuracy are shown in figure 4.10. Colours in this figure correspond to mafic metavolcanics being displayed in green, Proterozoic diabase in magenta, felsic metavolcanics in orange, late post-volcanic intrusions in blue, and early syn-volcanic intrusions in red. The early syn-volcanic sedimentary rocks are not shown in this region of the map.

Table 4.3: Confusion matrix for EnMap (Aggregation PSF) MF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	79.25	0	0	0	0	10.78	15.02	34.38	20.75	79.25	65.63
Pd	1.89	100	2.5	0	0	0.49	14.55	4.84	0	100	95.16
Esvi	0	0	97.5	0	10.34	6.86	13.85	33.9	2.5	97.5	66.1
Esvs	0	0	0	75	10.34	0	3.52	40	25	75	60
Fmv	0	0	0	25	79.31	3.43	13.15	17.86	20.69	79.31	82.14
Mmv	18.87	0	0	0	0	78.43	39.91	5.88	21.57	78.43	94.12
Total	100.01	100	100	100	100	100	100				
Overall Accuracy:		**83.33%**									
Kappa:		0.7742									

Table 4.4: Confusion matrix for EnMap (Gaussian PSF) MF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	78.18	0	2.5	0	0	11.33	15.69	35.82	21.82	78.18	64.18
Pd	1.82	100	0	0	1.82	0.99	15.22	6.15	0	100	93.85
Esvi	0	0	95	0	12.73	8.37	14.52	38.71	5	95	61.29
Esvs	0	0	0	84.62	9.09	0	3.75	31.25	15.38	84.62	68.75
Fmv	1.82	0	2.5	15.38	76.36	2.46	11.94	17.65	23.64	76.36	82.35
Mmv	18.18	0	0	0	0	76.85	38.88	6.02	23.15	76.85	93.98
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**82.20%**									
Kappa:		0.7605									

Table 4.5: Confusion matrix for EnMap (Sinc-Hamming PSF) MF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	77.36	0	2.63	0	0	10.24	14.89	34.92	22.64	77.36	65.08
Pd	0	100	0	0	0	1.95	14.89	6.35	0	100	93.65
Esvi	0	0	81.58	0	10.53	3.9	10.64	31.11	18.42	81.58	68.89
Esvs	0	0	0	72.73	8.77	0	3.07	38.46	27.27	72.73	61.54
Fmv	1.89	0	15.79	27.27	80.7	4.39	15.37	29.23	19.3	80.7	70.77
Mmv	20.75	0	0	0	0	79.51	41.13	6.32	20.49	79.51	93.68
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**82.27%**									
Kappa:		0.7572									

Table 4.6: Confusion matrix for ProSpecTIR-SPECIM MF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	76.94	0.02	1.86	0.04	0.06	14.52	14.96	46.03	23.06	76.94	53.97
Pd	3	94.07	1.79	8.16	1.16	6.33	17.53	22.28	5.93	94.07	77.72
Esvi	0.54	2.96	92.31	13.21	20.25	11.18	16.52	58.74	7.69	92.31	41.26
Esvs	1.15	2.56	1.97	75.43	23.28	4.01	9.5	68.73	24.57	75.43	31.27
Fmv	0	0	2.08	3.02	53.17	6.88	12.67	27.34	46.83	53.17	72.66
Mmv	18.37	0.39	0	0.12	2.08	57.08	28.83	8.15	42.92	57.08	91.85
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**67.17%**									
Kappa:		0.583									

Table 4.7: Confusion matrix for ProSpecTIR-SPECIM MF classification results with 17x17 spatial filtering applied.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	77.67	0	3.23	0	0	14.22	14.98	45.61	22.33	77.67	54.39
Pd	0.4	99.1	0.66	0.95	0	5.47	17.02	57.18	8.78	91.22	42.82
Esvi	0	0	92.42	6.46	18.55	11.17	15.47	22.29	39.05	60.95	77.71
Esvs	0	0.9	3.69	91.22	18.53	2.55	8.38	8.65	39.82	60.18	91.35
Fmv	0	0	0	1.37	60.95	6.41	13.58	15.67	0.9	99.1	84.33
Mmv	21.93	0	0	0	1.98	60.18	30.56	55.9	7.58	92.42	44.1
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**71.39%**									
Kappa:		0.633									

4.5.4 Random Forest Classification

The RF classification results from both the ProSpecTIR–SPECIM and EnMap imagery (figure 4.10) are similar to those found in the MF results however classification accuracy rates and visual inspection indicate this classification is slightly lower performing for the EnMap data and higher for the ProSPecTIR although units appear to be slightly more homogenous than the MF results for the ProSpecTIR–SPECIM data. Erroneous pixels are clearly visible in the upper left corner of figure 4.10 where the early syn-volcanic intrusions and felsic metavolcanics (in red and orange respectively) are classified as mafic metavolcanics (in green) particularly more so in the results based on the ProSPecTIR-SPECIM imagery.

For the EnMap data this resulted in an overall higher accuracy of 76.2% (*k*: 0.67) for the Gaussian PSF. Class accuracies ranged from 61.9 to 83.8% for the early syn-volcanic intrusions and Proterozoic diabase respectively. And for the airborne data the overall accuracy was 69.2% (*k*: 0.57) with class accuracies ranged from 40.5% to 89.6% for the felsic metavolcanics and Proterozoic diabase respectively.

With the RF classification algorithm, class and overall accuracy is dependent on the number of trees grown. The larger the number of trees grown the greater the improvement in accuracy however a maximum benefit is eventually reached. Plotted accuracy against the number of trees grown for each and all classes is shown in figures 4.12 and 4.13. In these figures the error rates are separated by their class and shown as dashed lines: late post volcanic intrusions (red), mafic meta-volcanic (black), early, syn-volcanic intrusions (magenta), felsic meta-volcanic (blue), Proterozoic diabase (green),

and early, syn-volcanic sedimentary rocks (cyan). The solid line indicates the overall classification error rate. Using the EnMap imagery, for all of the classes the rate fluctuated (caused by a relatively lower number of verification pixels), eventually stabilized and plateaued at ~150 trees indicating the maximum accuracy with this number of trees. For the ProSpecTIR–SPECIM data the number of trees was also ~150 with the exception of the Proterozoic diabase (consisting of the lowest error rate) that maximized at ~200. Any additional trees were not required and added unnecessarily computation time.

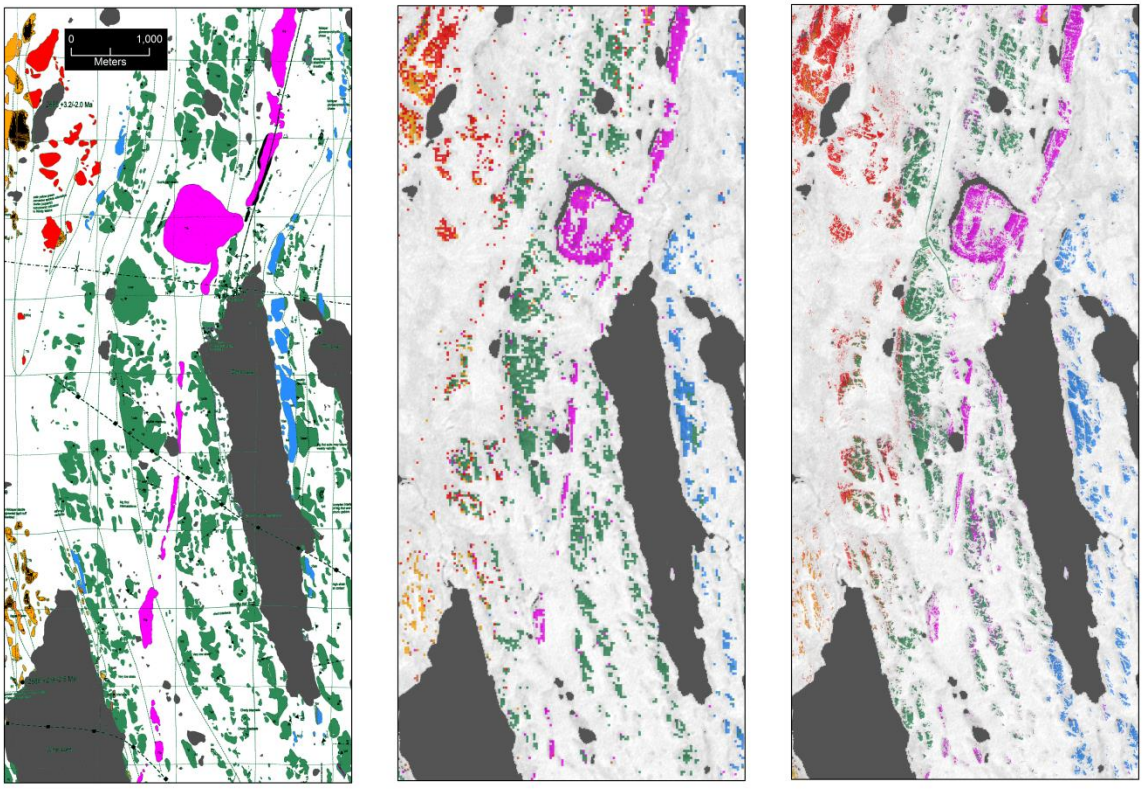

Figure 4.9: Comparison of the generalized published geology map (left) to that of the spatially filtered MF classification results for the simulated EnMap (Sinc-Hamming PSF) scene (center) and ProSpecTIR–SPECIM data (right). Map scale is roughly 1:100,000.

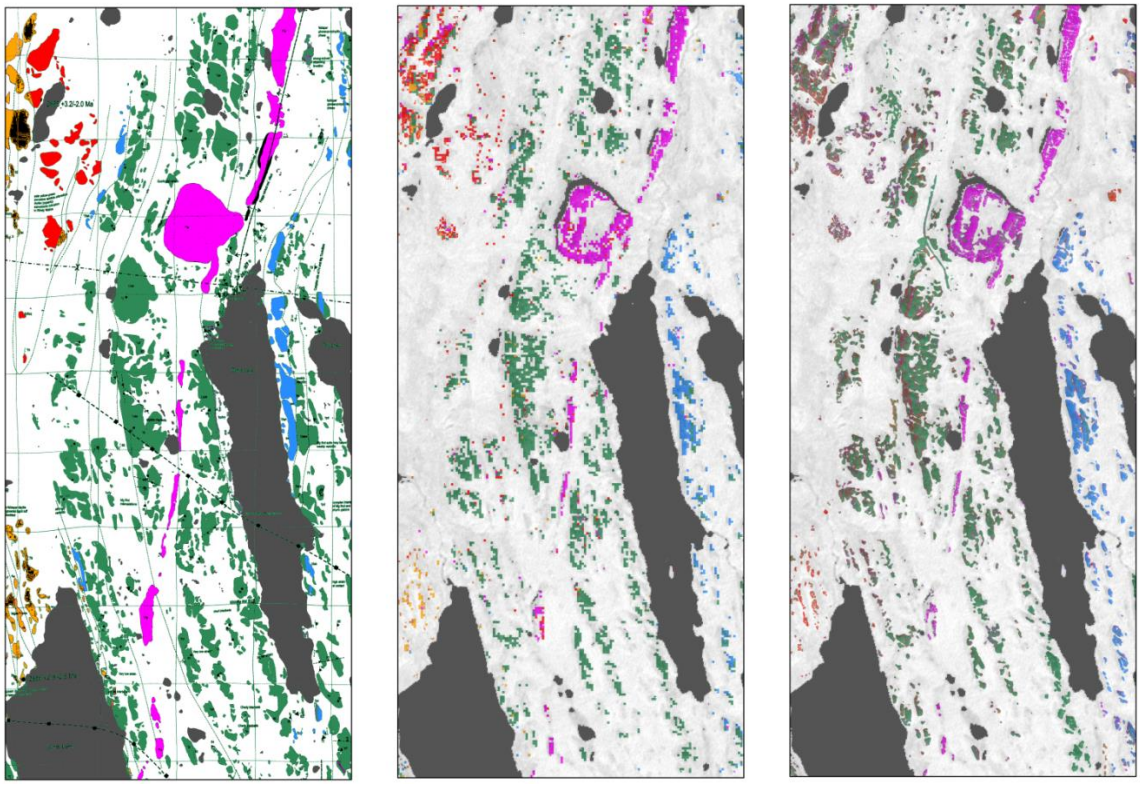

Figure 4.10: Comparison of the generalized published geology map (left) to that of the RF classification results for the simulated EnMap (Sinc-Hamming PSF) scene (center) and ProSpecTIR–SPECIM data (right). Map scale is roughly 1:100,000.

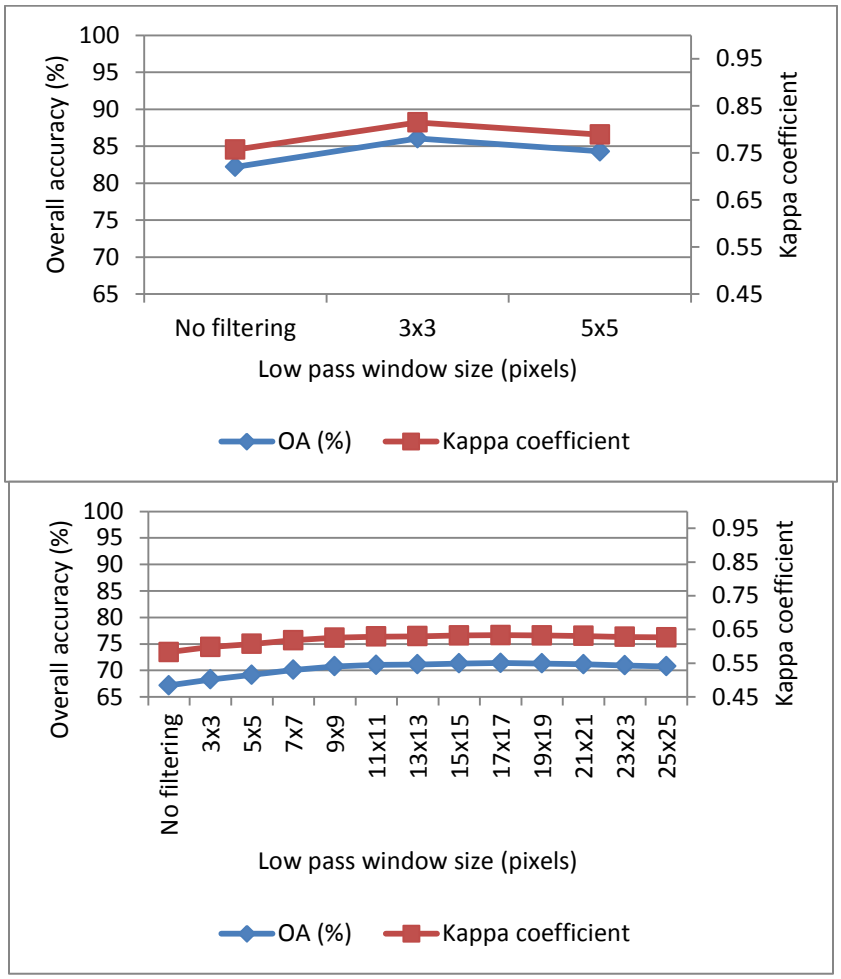

Figure 4.11: Graph of the changing overall accuracies and kappa coefficients with differing filtered window size for the classified maps based on Matched Filtering scores. At top are the values for the ProSpecTIR–SPECIM data and at bottom are the simulated EnMap (u sing a Sinc-Hamming PSF) results.

Table 4.8: Confusion matrix for EnMap (Aggregation PSF) RF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	67.74	1.47	0	0	0	9.72	13.01	37.31	32.26	67.74	62.69
Pd	4.84	86.76	7.14	25	6.25	3.24	15.92	28.05	13.24	86.76	71.95
Esvi	0	10.29	52.38	6.25	8.75	0.81	7.57	43.59	47.62	52.38	56.41
Esvs	0	1.47	9.52	56.25	11.25	0.81	4.85	64	43.75	56.25	36
Fmv	0	0	4.76	12.5	45	2.83	9.13	23.4	55	45	76.6
Mmv	27.42	0	26.19	0	28.75	82.59	49.51	20	17.41	82.59	80
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**72.23%**									
Kappa:		0.6056									

Table 4.9: Confusion matrix for EnMap (Gaussian PSF) RF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	72.58	1.47	0	0	0	8.91	13.2	33.82	27.42	33.82	27.42
Pd	3.23	83.82	4.76	12.5	3.75	1.21	13.4	17.39	16.18	17.39	16.18
Esvi	4.84	10.29	61.9	0	10	1.62	9.32	45.83	38.1	45.83	38.1
Esvs	0	2.94	7.14	62.5	6.25	0.81	4.27	54.55	37.5	54.55	37.5
Fmv	0	1.47	11.9	25	62.5	4.45	13.79	29.58	37.5	29.58	37.5
Mmv	19.35	0	14.29	0	17.5	83	46.02	13.5	17	13.5	17
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**76.31%**									
Kappa:		0.6688									

Table 4.10: Confusion matrix for EnMap (Sinc-Hamming PSF) RF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	72.58	1.47	0	0	1.25	8.91	13.4	34.78	27.42	72.58	65.22
Pd	4.84	83.82	11.9	25	5	2.83	15.53	28.75	16.18	83.82	71.25
Esvi	6.45	11.76	52.38	6.25	6.25	0.4	7.96	46.34	47.62	52.38	53.66
Esvs	0	2.94	4.76	62.5	10	0.81	4.66	58.33	37.5	62.5	41.67
Fmv	0	0	2.38	6.25	46.25	1.62	8.35	13.95	53.75	46.25	86.05
Mmv	16.13	0	28.57	0	31.25	85.43	50.1	18.22	14.57	85.43	81.78
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**74.19%**									
Kappa:		0.6322									

Table 4.11: Confusion matrix for ProSpecTIR-SPECIM RF classification results.

Class	Lpvi	Pd	Esvi	Esvs	Fmv	Mmv	Total	Commission Accuracy %	Omission Accuracy %	Producer's Accuracy %	User's Accuracy %
Lpvi	82.24	0.24	2.15	0.43	2.98	14.23	18.01	43.96	17.76	82.24	56.04
Pd	5.88	89.59	17.38	8.49	1.62	2.27	16.17	20	10.41	89.59	80
Esvi	1.76	7.88	46.27	6.52	6.74	3.74	6.72	65.45	53.73	46.27	34.55
Esvs	0.33	1.86	11.97	72.56	23.55	1.1	6.82	68.82	27.44	72.56	31.18
Fmv	2.46	0.19	14.97	11.66	40.47	8.57	11.38	51.45	59.53	40.47	48.55
Mmv	7.33	0.23	7.26	0.34	24.64	70.1	40.91	11.41	29.9	70.1	88.59
Total	100	100	100	100	100	100	100				
Overall Accuracy:		**69.24%**									
Kappa:		0.574									

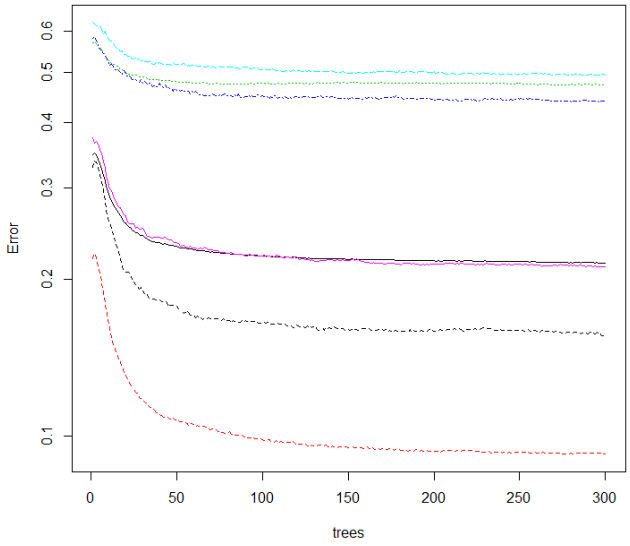

Figure 4.12: The error rate with the number of trees grown in the RF classification for the ProSpecTIR – SPECIM derived ratios and indices.

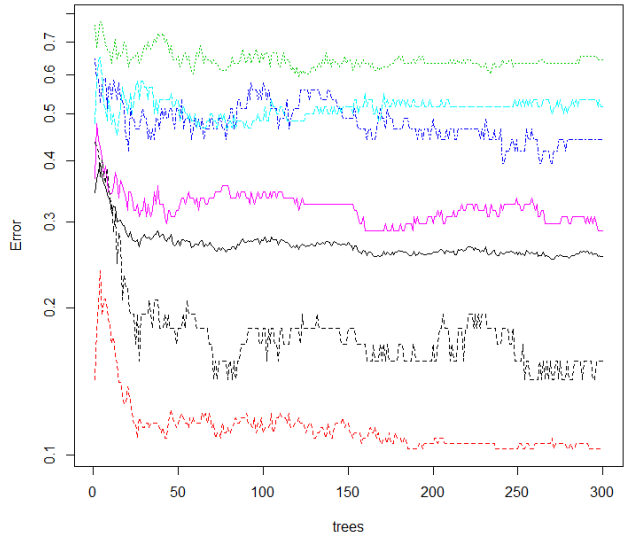

Figure 4.13: The error rate with the number of trees grown in the RF classification for the EnMap derived ratios and indices.

4.5.5 Differences in modelled Point Spread Function (PSF)

Table 4.12 displays the overall accuracies of the MF classification using the different modelled PSFs before the spatial filtering was applied. Table 4.13 similarly shows accuracies for the RF classification. There are only slight differences in the exactness of the predicted results that relates to the inward weighting of the pixels. For the MF classification the pixel aggregation PSF achieves the highest accuracy whereas the Gaussian filter reduces this rate by just 1.1%. The results of the RF classification show a conversely different pattern; The Gaussian PSF achieved the highest results over pixel aggregation (a difference of 4.1%). Given that satellite sensors measure radiance within the pixel dimensions in a non-uniform manner (the instantaneous field of view (IFOV)), where the central part of the pixel is weighted more heavily, pixel aggregation although it is the least unrealistic, contains only a minor difference.

The largest difference between PSFs in the confusion matrices appears to be between class accuracies where in the most extreme case there is a 17.5% difference for the felsic metavolcanics between the Gaussian and pixel aggregation PSFs during the RF classification.

Table 4.12: MF Classification accuracies as a function of the modelled PSF

PSF Shape	Overall accuracy (%)
Gaussian	82.20
Pixel Aggregation (mean values)	82.27
Sinc - Hamming	83.33

Table 4.13: RF Classification accuracies as a function of the modelled PSF

PSF Shape	Overall accuracy (%)
Gaussian	76.31
Pixel Aggregation (mean values)	72.23
Sinc - Hamming	74.13

To evaluate the visual implications that each modelling window has on the dataset, and to compare with an existing hyperspectral satellite image, zoomed sections of a true colour image are compared to an existing Hyperion image (see figure 4.14). The Hyperion satellite has the same 30 m spatial resolution and assumed expected PSF shape (Liao et al., 2000) as the EnMap satellite. The modelling EnMap scripts were applied using a similar airborne AISA DUAL hyperspectral dataset (2 m spatial resolution) but for the Victoria, BC airport (YYJ). Given that the Gaussian modelled PSF more significantly weights the area farther outside the pixel dimensions as the sinc-hamming PSF (while still maintaining the same GIFOV) it blurs the central values more than the other two methods which in contrast sharpen the image. Compared to the Hyperion image, shown at the top in the figure, the Gaussian PSF appears to be most similar. Although there is a contrast in the state of vegetation growth between images the differences can be seen when looking at the edges of the runway tarmac and the painted lines.

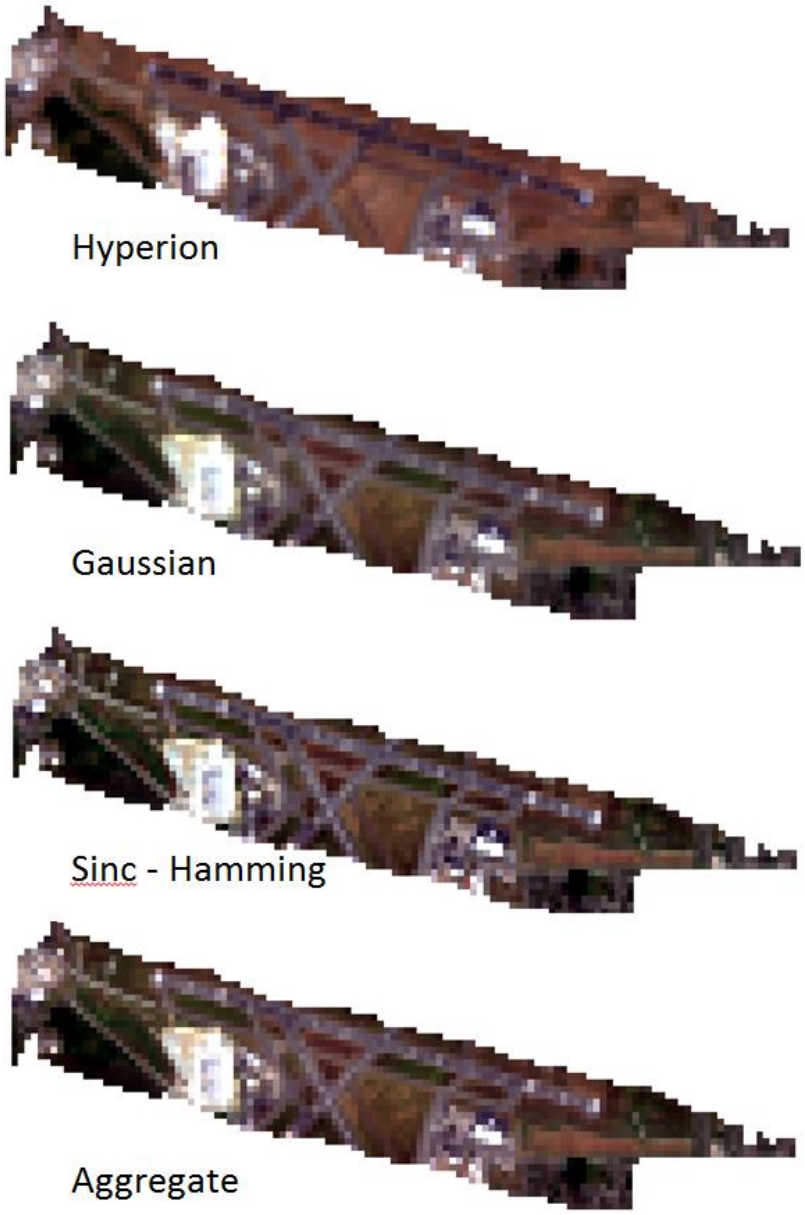

Figure 4.14: True colour images showing the visual differences between the EO-1 Hyperion sensor imagery (July 2013), EnMap simulated from airborne data (July, 2014) using different modelled PSFs. Map scale is roughly 1:35,000.

4.6 Discussion

The relationships of spectral and spatial variability to accuracy are highlighted in this study. When sufficient masking of the non-geological content (mainly vegetation and

water) is applied, lithological units exhibit a high degree of class spectral variability, significantly so that pixel mixing with other non-geological content found in coarser resolution data is less degrading to the statistics of the modelled results. Since class spectral variability are sufficiently high, coarser resolution or spatial filtering reduced this variability and increased the accuracy of classification. This can be seen in the comparative results of differing spatial resolutions and the spatial filtering. However given that rock outcrops have a patchy distribution, spatial variability can also be high. With the spatially filtered derived abundances and coarser resolution the results look more like a thematic map yet there is a loss of detail and correctly mapped smaller occurrences of outcrops – either through misclassification and/or oversimplification (see figure 4.15).

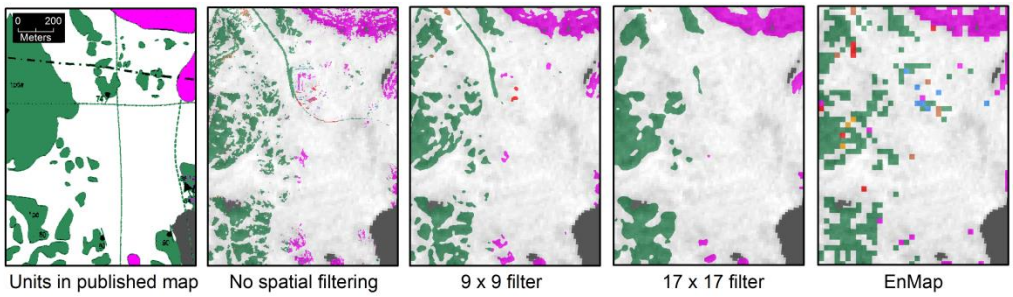

Figure 4.15: Visual loss and simplification of the classification when spatially filtered MF results are applied through increasing kernel sized spatial filters. EnMap classifcation results are also shown for comparison. Map scale is roughly 1:40,000.

The classification approach utilized is also related to spectral and spatial characteristics of the mapped phenomena. RF classification accuracies using ratios were relatively higher for the airborne data compared to the MF results but lower for the EnMap data. This indicated that in the presence of low pixel mixing the ratio indices approach showed

improved discrimination of rocks. This same observation has been found by Woodcock and Strahler (1987) and Atkinson (1997) who identified that for high resolution imagery, at pixel sizes that more than exceed the local spatial variance, a spectral (hard) classifier is more appropriate and conversely where it nearly matches or is coarser than the local variance, a spectral unmixing approach is needed. Atkinson (1997) further implies that if the spatial resolution is coarser than the spatial variability then the dataset should not be used for mapping.

To this end, semi-variogram analysis was conducted to determine the local spatial variance of the outcropping areas using the ProSPecTIR-SPECIM inverse MNF data (see chapter 2 for more information on the use of semi-variograms). As local variance can vary depending on the wavelength (Curran, 1988; Atkinson, 1997; Rahman et al., 2003) nine spectral bands and each individual band index were analyzed (see figure 4.16). In all cases, the variogram showed a similar pattern: a consistent logarithmic relationship of variance to distance with a range that is difficult to pinpoint but slightly greater than 80 m. There is also higher heterogeneity in the NIR wavelengths. This heterogeneity is indicated by a relatively higher sill in this spectral coverage (top plot of figure 4.16) and the spectral indices that rely on these wavelengths (bottom of figure 4.16). Given that the optimal spatial resolution is deemed as half the range the results of the filtered ProSpecTIR-SPECIM and EnMap MF classified maps agree well with this proposal.

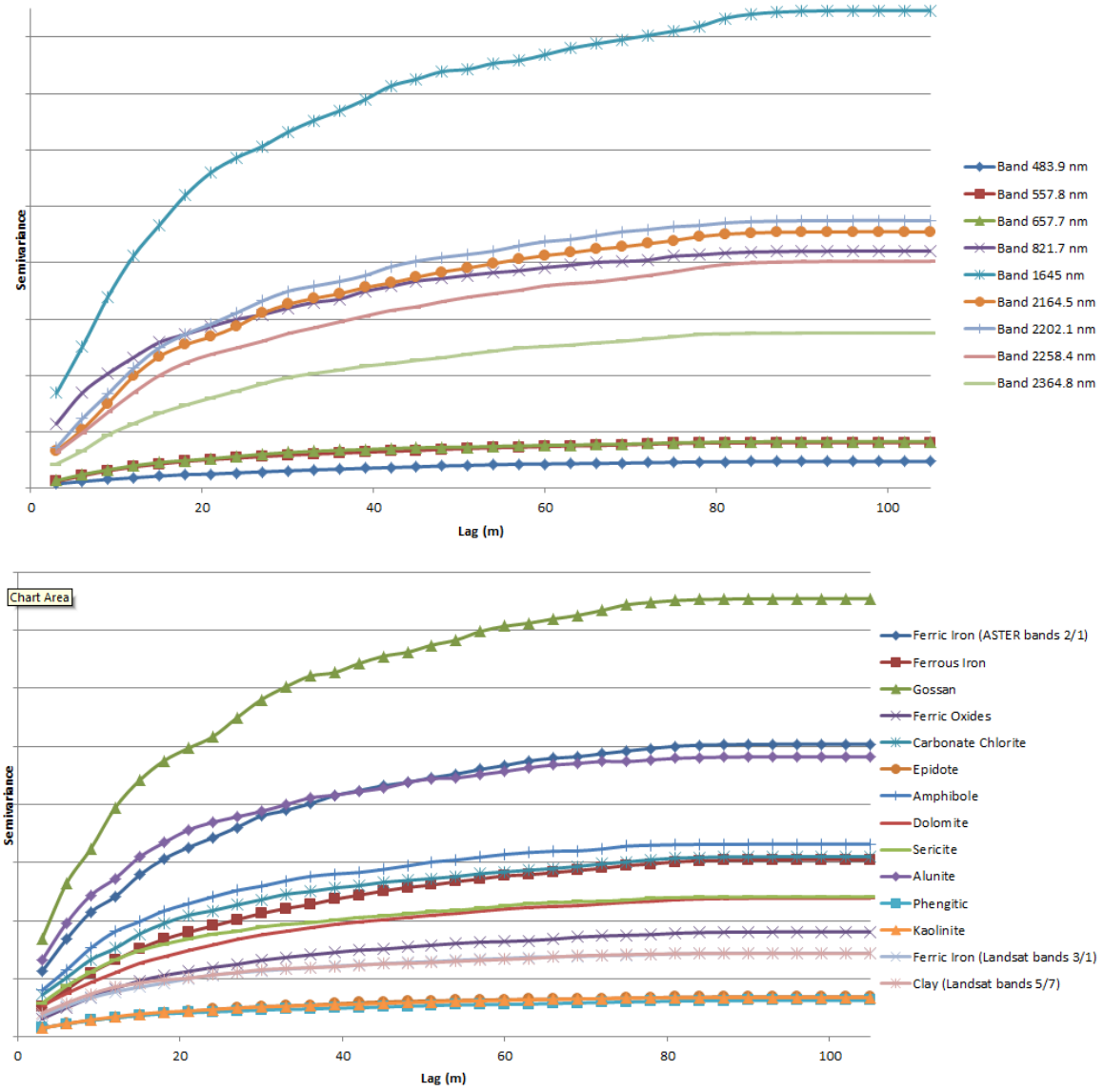

Figure 4.16: Semivariograms for five wavelength positions (top) and band ratio and indices (bottom) from the 3m ProSpecTIR - SPECIM image.

Fine spectral resolution (less than 6.5 nm VNIR and 10 nm SWIR) appeared to be less important than the spatial resolution since in neither classification approach did the higher spectral resolution of the ProSpecTIR-SPCIM data outperform the EnMap data. No spectral features were visually found in the ProSpecTIR-SPCIM spectral signatures that were not found in those of the EnMap data. However the spectral resolution and

coverage of both hyperspectral images clearly was advantageous as the MF classification was able to distinguish subtle spectral differences in lithological classes to a level that according to the kappa coefficient was *substantial*. The spectral resolution of both sensors was fine and contiguous enough to discriminate similar rock units with only minor spectral differences and unmix coarser pixels. In comparison, when the class spectral signatures are resampled to the geological multispectral satellites: ASTER and Landsat 8 (figure 4.17), their subtle features are lost and would otherwise be more challenging to classify.

Despite lower overall classification statistics associated with the RF classification this approach provided two advantages over the MF classification. First, although out of scope of this study, the RF classification, as a non-parametric classifier, can be used alongside other multi-sourced supplementary datasets such as Digital Elevation Models (DEMs) and/or airborne magnetics that have been shown when fused to increase lithological classification effectiveness (Schetselaar and de Kemp, 2000; Brown et al., 2007) .

Secondly, RF can provide insight into the importance of specific bands that otherwise was not available using MF. With the ordering of the variable importance and mean decrease of Gini index scores for the input bands (figure 4.18) the mineral absorption feature(s) that help to delineate class units can be observed. It is not surprising to discover that the top five important band ratios/indices for both sensors are designed and deemed appropriate for discriminating minerals from three separate key mineral classes: iron, silicates, and mafic rocks (Kalinowski & Oliver, 2003).

Insight into the delineating mineral composition associated with the mafic/felsic rocks was also found using these variable importance bands/indices. For example, with a linearly stretched colour RGB (red, green, blue) composite of the three highest variable importance band (figure 4.19) notable differences can be observed in the mineral absorption features between spectral units.

The kaolinite index (shown in the green channel), a mineral derived from weathering feldspar, helps to distinguish the mafic from the felsic rocks that is otherwise difficult to accomplish. In figure 4.19 these rocks appear as pixels of deep red and light magenta respectively. The gossan ratio band (blue channel) helps to distinguish the Proterozoic diabase and post-volcanic intrusions in this area to stand out as cyan and dark royal blue colours respectively in the colour composite. The early, syn-volcanic intrusions appear as magenta pixels with associated higher index values in the ferric iron and kaolinite images and low values in the gossan image.

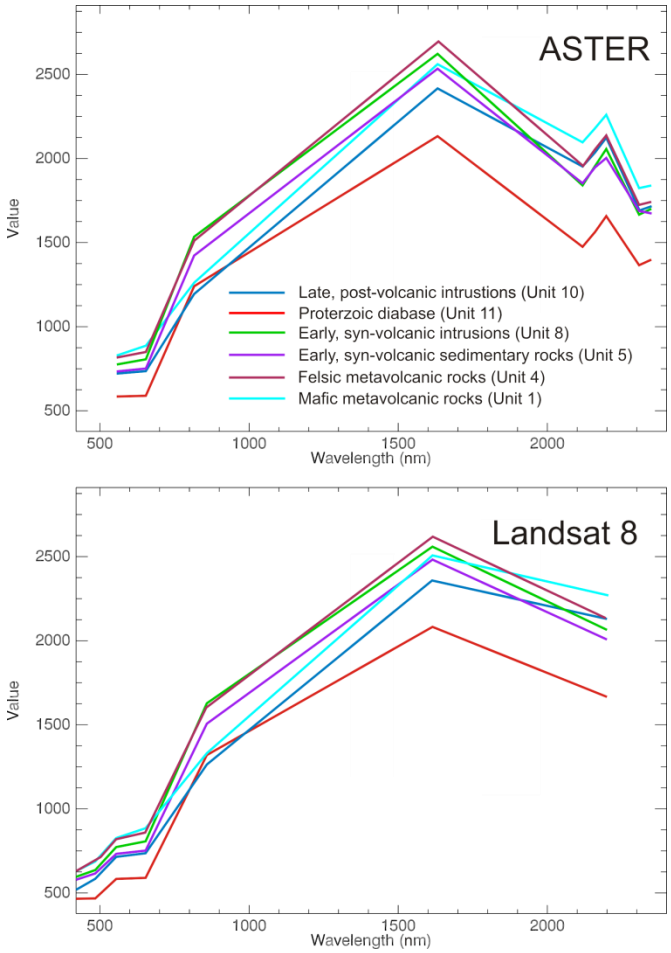

Figure 4.17: Class mean spectral signatures as they appear in the multispectral sensors ASTER and Landsat 8.

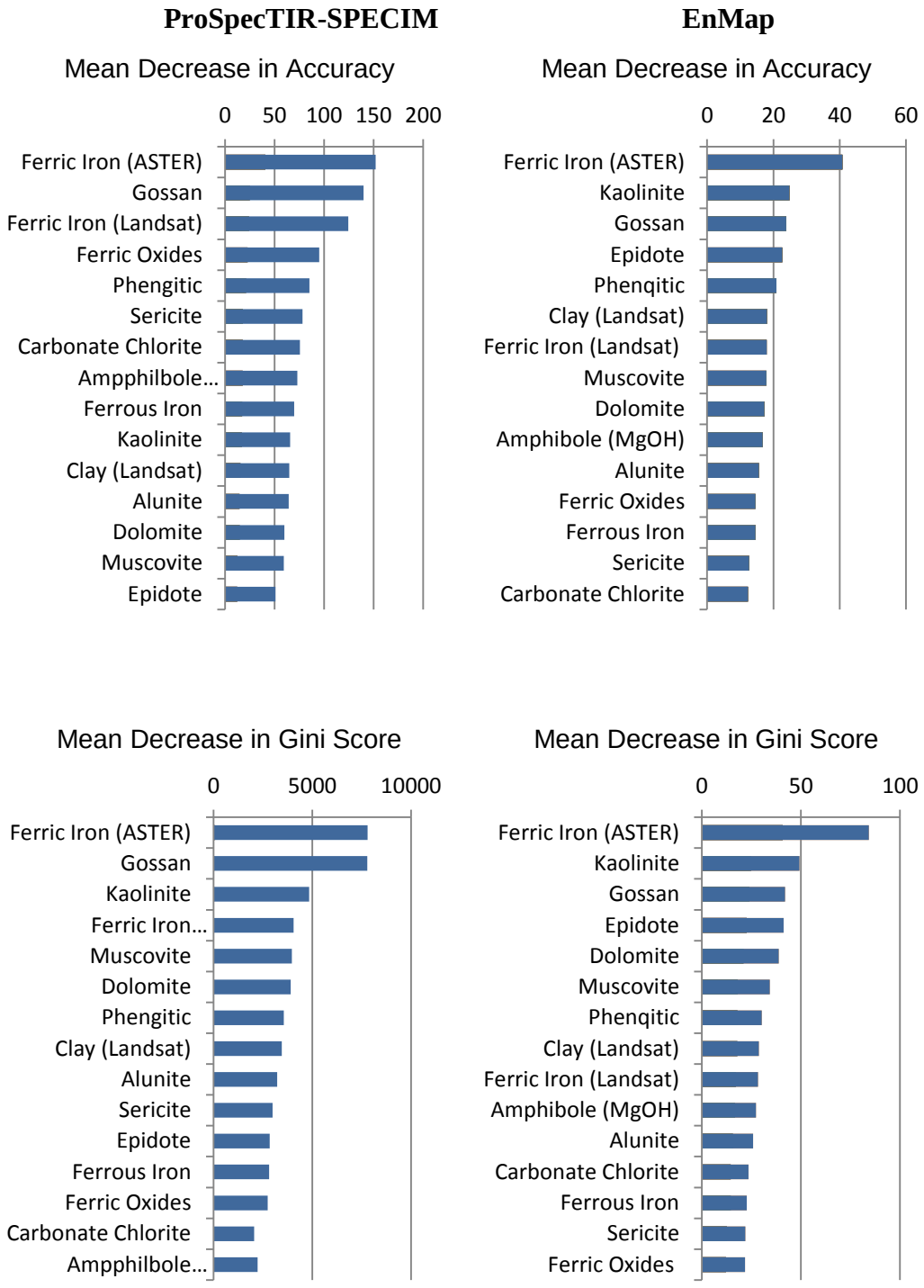

Figure 4.18: Variable importance plots showing the mean decrease in accuracy (top) and Gini scores (right) from the RF classification of the ProSPecTIR – SPECIM (left) and the EnMap imagery (right).

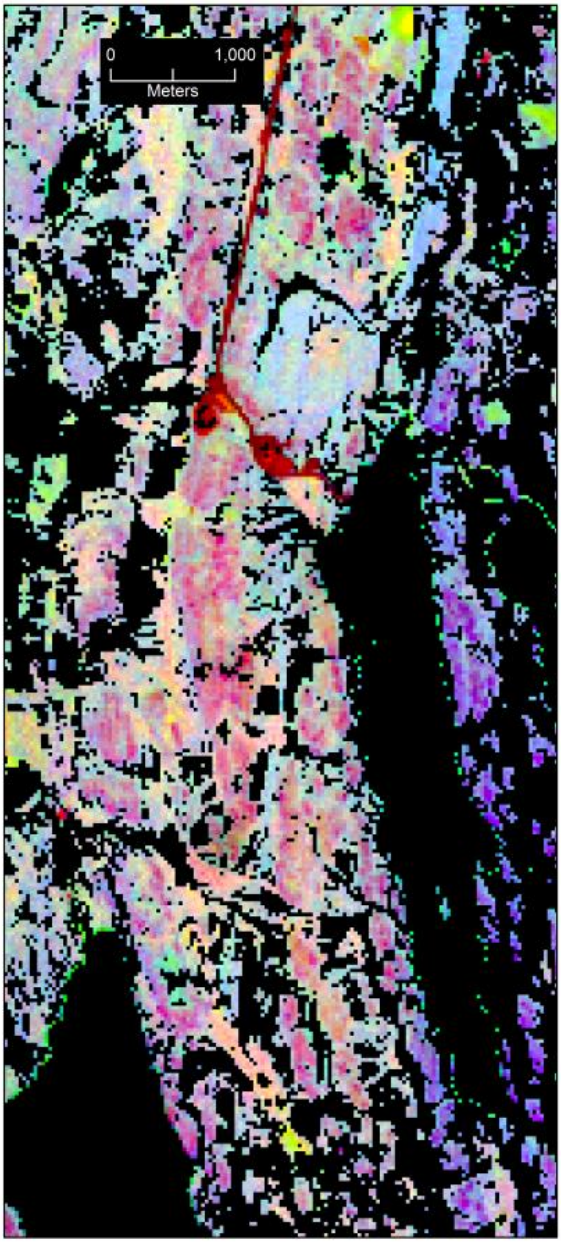

Figure 4.19: RGB composite of the three highest variable importance bands (R: Ferric Iron, G: Kaolinite, and B: Gossan) from the RF classification (EnMap) before thresholding probability values. Areas shown in black represent masked vegatation and water.

4.7 Conclusion

The primary aim of this chapter was to access and compare the performance of EnMap and ProSpecTIR-SPICM in mapping lithological units for producing spectral maps. Both were successful with EnMap discriminating geology units more successful compared to the airborne data. The results here indicate that its high contiguous spectral and moderate spatial resolution will be able to handle the spectral variability associated with mixed lithologies. Higher spatial resolution imagery would facilitate the spatial identification of smaller scale features however it came at a distinct cost of lowered accuracy.

MF filtering was the more appropriate classification mechanism for the coarser EnMap data but also appropriate for the higher resolution airborne data when spatial filtering of the MF scores was applied. Conversely, RF classification was found to be a viable classification approach for the higher resolution airborne data.

Visual comparisons between the classified outputs and the published map highlighted the significance of cartographic thematic representation of the outcropping areas on differences in mapped results and may contribute to error captured in the confusion matrices. Geologic units are often simplified by the mapping geologist such that units are either generalized or mapped in areas not necessarily exposed and thus are not represented in the classified map. Further visual investigation of the true colour high resolution imagery and oblique aerial photographs revealed that several areas are indeed not outcropped and are generalized through geologic interpretation by the geologist. Such representation is visible in the central part of figure 4.10 and 4.11 where the large

Proterozoic diabase dyke (in magenta) is a large body in the geologic map but only the exposed sections are displayed in the classified map.

Geological mapping is complicated by the subjective geologist interpretation and simplification to produce a published map. The biophysical environment and the variance of outcrop sizes and lithological units additionally may not match the scale that is being mapped. Although ideally any mapping activity is done at a scale that is appropriate to this variation, the geologist is still required to simplify the information onto a map at a given scale. Further, inaccessible areas because of steep terrain, dense vegetation, water, or marshy land may produce a patchy distribution in visited sites during fieldwork. The use of hyperspectral imagery provides a useful tool for helping with producing these thematic maps as the imagery provides useful information that can be generalized (through spatial filtering) to the appropriate scale prior to and after geological fieldwork.

Chapter 5: Conclusion and Recommendations

The ongoing development of hyperspectral imaging (HSI) technology will lower the current high acquisition costs of airborne surveys in remote locations and increasingly provide access to available data for mapping projects. The Canadian Arctic, with the widespread exposure of bedrock, is an ideal environment for geological mapping using these sensors. It provides a significant advantage to geological mapping programs given the high costs associated with conducting fieldwork in such an environment; any additional supporting information for a region will streamline fieldwork by providing knowledge of which areas require additional detailed work. The acquisition of airborne hyperspectral imagery provides the advantage of high spatial and spectral resolution and SNR as well as greater control over acquisition conditions (sun illumination and meteorological conditions). However these surveys can be prohibitively expensive. With the development of new satellite sensors that now have higher SNR than the previously launched hyperspectral satellites, with global coverage, and with free data accessibility they will imaginably be widely used for regional geological mapping. For the Canadian Arctic the use of HSI satellite technology may have a profound impact on the ability to create accurate predictive maps and influence the approach taken for future geological mapping.

The first and second objectives of this study were to investigate if imagery acquired from the soon to be launched EnMap satellite and a high resolution airborne dataset could map lithological units and what would the differences be between these sensors? It found that both sensors can be used to produce valid predictive lithological maps for this type of region that is rich in feldspar, silicates, and quartz, minerals that lack pronounced

distinguishing spectral absorption features in the VNIR and SWIR wavelengths and thus are challenging to delineate for imaging sensors. The predictions based on subtle absorption features using MF abundance maps matched the published map reasonably well. The locations of mapped units agreed with the dominant lithological units of the region and produced statistically accurate maps but there is a loss of spatial detail; the high resolution airborne imagery with an associated high spectral variability produced less spatially cohesive mapping but improved upon when the abundance images were spatially filtered with a large kernel size. Similarly, the EnMap MF abundance maps were also filtered using a small windowed kernel size to increase mapping accuracy. The end result however meant that small occurrences of outcrops were lost or overly generalized.

Most notable in this study was the comparatively higher accuracy of the mapping results from the EnMap simulated data over the airborne imagery. Past hyperspectral remote sensing studies directed at geological mapping using spaceborne sensor resolutions have assumed that the accuracy decreases with decreasing resolutions (Kruse, 2003; Rogge el al., 2014) however this was without the use of any statistical classification metric. This study concludes that lower resolution becomes an advantage as it reduces the spectral variance of the internal class spectra of training areas and the imagery it attempts to classify. This lower spectral variance is additionally the result of the coarser PSF of the EnMap sensor acting as an averaging spatial filter on high spectrally variable classes. .

The optimal spatial resolution in remote sensing classification has previously been shown to not be the highest spatial resolution for a study area but is dictated by the relationship between the spatial patterns of targeted classes, in this case the lithological units, and the

resolution. The appropriate resolution is ideally half of the spatial dependence of the targeted features with distance (Woodcock et al., 1988; Rahman et al., 2002). Lithological units which contain higher internal variance of spectra necessitate coarser spatial resolution imagery to produce accurate maps (Cushnie, 1987). The result of semi-variogram analysis, the better accuracy found in the EnMap predictions, and the spatial filtering thereby suggests that the spatial dependence of the outcropping landscape in this study is greater than 80 m.

Caution however must be applied when using semi-variograms for detecting the optimal resolution as outcrops, unlike other landscape phenomena where semi-variance analysis has been used for the optimal pixel size, can vary greatly in their size, shape, and fragmentation. This means they are not relatively consistent in these characteristics as compared to say a forest of trees. When such values of localized spatial variance contain irregular normality and stationarity they present problems for the calculated variogram (Bohling, 2003) and therefore the optimal spatial resolution may require further consideration (Smith et al., 2002, 2003; Bøcher and McCloy, 2006). Additionally, with coarser resolution this influence of landscape structure heterogeneity reducing accuracy becomes more significant (Lin et al. 2008).

This relationship of spatial resolution and spectral variability of the landscape was also found to affect the performance of the RF classification using band ratios and indices. The RF classification generally showed visually better performance in maintaining the smaller occurrences of isolated unique bedrock classes. Confusion matrix analyses indicated however that the classification accuracy were actually lower than the MF

results for EnMap. Band ratios and indices may provide more useful information when the scale of mapping requires maintaining smaller isolated occurrences of class units as was the case with the airborne ProSpecTIR-SPECIM. Coarse spatial resolution of the EnMap data however restricts the ability to detect isolated, smaller, and thinner occurrences of outcrops. This can be problematic for smaller scale mapping and environments with significantly high degrees of pixel mixing.

The third objective of this thesis was to investigate the effect that different approaches to modelling a coarser sensor's PSFs during simulation might have on classification accuracy. Previous studies that looked at spatial resampling of imagery to assess geological mapping classification accuracies have not considered the influence that simple pixel aggregation might have on the classification (Staenz et al., 2001; Kruse et al, 2011). This study showed that overall and class accuracies vary between different modelled PSF and classification approaches. No consistent pattern was observed between PSFs as one modelled PSF approach outperformed another but the opposite occurred when the classification approach (RF versus MF) was changed. In this study, class accuracy varied by as much as 17% and overall by 4% between imagery modelled through these PSFs. Researchers considering differing PSFs can expect a similar statistical response as any smoothing of the data caused by a larger kernel window may or may not improve accuracy depending on the spatial dependence of the landscape under study and the classification approach used.

The assessment made in this thesis involved training and validation sites based on a 1:25,000 scaled map because this map afforded suitable detail to delineate outcrops.

When the scale of mapping is 1:50,000 or coarser, the 30 m resolution of the EnMap imagery will be a more practical tool than the high resolution airborne data. Single pixels of an EnMap image when printed at 1:50,000 measure as 0.6 mm squares (180 dpi). Such an output resolution would provide useful information for mapping however any larger scale would lack the finer scale dependent information needed. The size of the smallest isolated outcrop would no longer be twice the resolution required for it to be delineated and mapped (Tobler, 1987). Conversely, if the scale were to be coarser than 1:50,000 it would increase the probability that multiple EnMap scenes would be required to produce a predictive map for a given study area. Since Canadian federal and territorial geological survey agencies use the National Topographical System (NTS) to publish maps they use a scale of 1:50,000 for their more detailed regional mapping. Accordingly more than one scene will be needed to cover a typical map given the 30 km swath coverage of EnMap and similar planned hyperspectral satellites. Care must be taken then that multiple scenes should be acquired during similar dates and times to avoid changes in illumination and seasonal changes in vegetation. Additionally mosaicking and radiometric balancing across the region or alternatively segregated classifications will add complications and processing time.

This thesis provides a positive outlook on the potential of the EnMap sensor. Spectral information will provide one additional source: detailed spectral reflectance, to a multitude of possible sources of information that mapping geologists can use to produce a multi-sensor predictive geologic map.

5.1 Suggestions for further research:

In this thesis, the specific ratios and indices used for the RF classification were chosen from previous multispectral studies where they have been proven to be effective. However, these ratios were designed for the measured broad wavelength coverage associated with multispectral sensors and don't consider possible fine narrow spectral features captured by a HSI sensor. Several mineral diagnostic features are known to exist with finer absorption features that could be utilized to produce greater accuracy results using hyperspectral data. Additionally, multispectral sensors only measure within specific spectral ranges whereas the continuous coverage of hyperspectral imaging provides the possibility that spectral information not measured by a multispectral sensor could provide helpful spectral features. It is probable that data mining methods (Guo, et al., 2006; Harris et al., 2006; Buddenbaum, 2014) could be used to statistically assess all possible band combinations to reveal the ideal ratios or bands for hyperspectral discrimination on the unmasked areas rather than relying on existing multispectral ratios and indices. Future studies which investigate or incorporate these methods to select the best band combinations would lead to increased lithological class separability and classification accuracy such that they could outperform MF results even at coarser scales.

Future studies should consider further effort to mask the overburden. Although unmasked in this study because it would be challenging to delineate such deposits, the spectral similarity of this surface and the scarcity of the early syn-volcanic sedimentary lithology unit resulted in this class being over estimated. Such misinformation would confuse the field geologists rather than support their work.

In this study, only the expected SNRs of the EnMap sensor were used in the creation of a synthesized scene. Acquiring an image where such required illuminations exist will be challenging in the Canadian North where such a required sun angle is rare. This is important since lower sun angles have previously been a source of classification errors as they decrease SNR and increase the length of shadows of scene objects (Harris et al., 2014). Continued work could involve simulating systematic decreases in SNR and measuring the decreased accuracy. Such work could be taken using the scripts written for this thesis.

Ongoing research could also consider comparing the capability of the EnMap satellite with the latest VNIR/SWIR multispectral sensor: Sentinel-2A (S2A). S2A was launched in June 23, 2015 and is unique compared to other multispectral sensors in that it provides 7 VNIR and 2 SWIR bands at 10 and 20 m GIFOV respectively and are positioned on wavelength ranges known to contain spectral absorption features of common alteration minerals such as calcite, alunite, jarosite sericite, kaolinite, dickite, and illite. Its higher spatial resolution and relatively high spectral resolution (compared to other multispectral satellites) may provide the finite resolution for discriminating smaller occurrences of exposed outcrops although as demonstrated in this study at the potential cost of accuracy.

Further investigations could also be conducted on the fusion of EnMap data with other data sources. The spatial resolution of EnMap provides the possibility to easily fuse this imagery with CDED topographic datasets, RADARSAT imagery, and other multispectral sensor data (i.e. Landsat and ASTER) that are acquired at this same resolution.

Geological studies involving data fusion with other available complimentary datasets are widely known to improve the predictive performance of supervised classification.

References

Achard, V., & Lenot, X. (2009). Atmospheric and topographic corrections for hyperspectral imagery. *First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS '09)* (pp. 1-4).

Acito, N., Diani, M., & Corsini, G. (2011). Subspace-based striping noise reduction in hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing, 49*(4), 1325-1342.

Ager, C. M., & Milton, N. M. (1987). Spectral reflectance of lichens and their effects on the reflectance of rock substrates. *Geophysics,* 52(7), 898-906.

Alparone, L., Selva, M., Aiazzi, B., Baronti, S., Butera, F., & Chiarantini, L. (2009). Signal-dependent noise modelling and estimation of new-generation imaging spectrometers. *First Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS '09)* (pp. 1-4).

Arai, K. (2013). Nonlinear mixing model of mixed pixels in remote sensing satellite images taking into account landscape. *International Journal of Advanced Computer Science and Applications, 4*(1), 1715–1723.

Atkinson, P. M. (1997). Selecting the spatial resolution of airborne MSS imagery for small-scale agricultural mapping. *International Journal of Remote Sensing, 18*(9), 1903-1917.

Balaguer-Beser, A., Ruiz, L. A., Hermosilla, T., & Recio, J. A. (2013). Using semivariogram indices to analyse heterogeneity in spatial patterns in remotely sensed images. *Computers & Geosciences, 50*, 115-127.

Barducci, A., Guzzi, D., Lastri, C., Marcoionni, P., Nardino, V., & Pippi, I. (2012). Simulating the performance of the hyperspectral payload of the PRISMA mission. *Geoscience and Remote Sensing Symposium (IGARSS), 2012 IEEE International (*pp. 5013-5016).

Barnsley, M. J., & Kay, S. A. (1990). The relationship between sensor geometry, vegetation-canopy geometry and image variance. *International Journal of Remote Sensing, 11*(6), 1075-1083.

Behnia, P., Harris, J. R, Rainbird, R., Williamson, M., & Sheshpari, M. (2012). Remote predictive mapping of bedrock geology using image classification of Landsat and SPOT data, western Minto Inlier, Victoria Island, Northwest Territories, Canada. *International Journal of Remote Sensing, 33*(21), 6876-6903.

Beletic, J. W., Blank, R., Gulbransen, D., Lee, D., Loose, M., Piquette, E. C., et al. (2008). Teledyne imaging sensors: Infrared imaging technologies for astronomy & civil space. *Proceedings of SPIE – The International Society for Optical Engineering, 7021* (pp. 70210H-70210H).

Bian, L., & Butler, R. (1999). Comparing effects of aggregation methods on statistical and spatial properties of simulated spatial data. *Photogrammetric Engineering and Remote Sensing, 65*, 73-84.

Blonski, S., Cao, C., Gasser, J., Ryan, R., Zanoni, V., & Stanley, T. (2000). Satellite hyperspectral imaging simulation. *Proceedings of the International Symposium on Spectral Sensing Research (ISSSR) 1999,* Las Vegas, USA, 1-4 November 1999.

Bøcher, P., & McCloy, K. R. (2006). The fundamentals of average local variance-part II: Sampling simple regular patterns with optical imagery. *IEEE Transactions on Image Processing, 15*(2), 311-318.

Bohling, G. (2005). Introduction to geostatistics and variogram analysis. *Kansas Geological Survey*, 1-20. Retrieved from: www.ecst.csuchico.edu.

Boncelet, C. (2005). Image noise models. *Handbook of image and video processing*, 325-335. London, UK: Elsevier Academic Press

Börner, A., Wiest, L., Keller, P., Reulke, R., Richter, R., Schaepman, M., et al. (2001). SENSOR: A tool for the simulation of hyperspectral remote sensing systems. *ISPRS Journal of Photogrammetry and Remote Sensing, 55*(5), 299-312.

Briottet, X., Marion, R., Carrere, V., Jacquemoud, S., Chevrel, S., Prastault, P., et al. (2011). HYPXIM: A new hyperspectral sensor combining Science/Defence applications. *Third Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS) (*pp. 1-4).

Brown, O., Harris, J. R., Utting, D., & Little, E. (2007). Remote predictive mapping of surficial materials on northern Baffin Island: Developing and testing techniques using Landsat TM and digital elevation data. *Geological Survey of Canada, Current Research,* 2007-B1.

Bruce, C. (2012). HyspIRI mission concept overview and recent ICE and TRL activities. *Proceedings HyspIRI Science Workshop,* Washington, DC.

Buckingham, R., & Staenz, K. (2008). Review of current and planned civilian space hyperspectral sensors for EO. *Canadian Journal of Remote Sensing, 34*(S1), S187-S197.

Budkewitsch, P., Staenz, K., Neville, R., Rencz, A., & Sangster, D. (2000). Spectral signatures of carbonate rocks surrounding the nanisivik MVT Zn-Pb mine and implications of hyperspectral imaging for exploration in Arctic environments. *Ore Deposit Workshop: New Ideas for a New Millennium, Technical Volume,* Cranbrook, Canada (pp. 5-6).

Burazerović, D., Heylen, R., Geens, B., Sterckx, S., & Scheunders, P. (2013). Detecting the adjacency effect in hyperspectral imagery with spectral unmixing techniques. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, 6*(3), 1070-1078.

Ceamanos, X., & Douté, S. (2010). Spectral smile correction of CRISM/MRO hyperspectral images. *IEEE Transactions on Geoscience and Remote Sensing, 48*(11), 3951-3959.

Chan, R. H., Ho, C., & Nikolova, M. (2005). Salt-and-pepper noise removal by median-type noise detectors and detail-preserving regularization. *IEEE Transactions on Image Processing, 14*(10), 1479-1485.

Chen, D., Stow, D., & Gong, P. (2004). Examining the effect of spatial resolution and texture window size on classification accuracy: An urban environment case. *International Journal of Remote Sensing, 25*(11), 2177-2192.

Chen, J., Blume, H., & Beyer, L. (2000). Weathering of rocks induced by lichen colonization—a review. *Catena*, *39*(2), 121-146.

Chen, Y., Ji, Y., Zhou, J., Chen, X., & Shen, W. (2012). Computation of signal-to-noise ratio of airborne hyperspectral imaging spectrometer. *2012 International Conference on Systems and Informatics (ICSAI) (*pp. 1046-1049).

Clark, R. (1999). Spectroscopy of rocks and minerals, and principles of spectroscopy. In Rencz, A. N. (Ed.), *Remote Sensing for the Earth Sciences: Manual of Remote Sensing*. 3rd ed. (pp. 3-58): New York, US: John Wiley & Sons.

Clark, R. N., Gallagher, A. J., & Swayze, G. A. (1990). Material absorption band depth mapping of imaging spectrometer data using a complete band shape least-squares fit with library reference spectra. *Proceedings of the Second Airborne Visible/Infrared Imaging Spectrometer (AVIRIS) Workshop* (pp. 176-186): Pasadena, US.

Coppo, P., Chiarantini, L., & Alparone, L. (2013). End-to-end image simulator for optical imaging systems: Equations and simulation examples. *Advances in Optical Technologies.*

Cota, S. A., Bell, J. T., Boucher, R. H., Dutton, T. E., Florio, C. J., Franz, G. A., ... & Paulson, D. B. (2010). PICASSO: an end-to-end image simulation tool for space and airborne imaging systems. *Journal of Applied Remote Sensing*, *4*(1), 1-36.

Cracknell, M. J., & Reading, A. M. (2014). Geological mapping using remote sensing data: A comparison of five machine learning algorithms, their response to variations in the spatial distribution of training data and the use of explicit spatial information. *Computers & Geosciences*, *63*, 22-33.

Crippen, R. E., & Blom, R. G. (2001). Unveiling the lithology of vegetated terrains in remotely sensed imagery. Unveiling the lithology of vegetated terrains in remotely sensed imagery. *Photogrammetric Engineering and Remote Sensing*, *67*(8), 935-943.

Curran, P. J. (1988). The semivariogram in remote sensing: an introduction. *Remote Sensing of Environment*, 24(3), 493-507.

Cushnie, J. L. (1987). The interactive effect of spatial resolution and degree of internal variability within land-cover types on classification accuracies. *International Journal of Remote Sensing*, 8(1), 15-29.

Dadon, A., Ben-Dor, E., & Karnieli, A. (2010). Use of derivative calculations and minimum noise fraction transform for detecting and correcting the spectral curvature effect (smile) in Hyperion images. *IEEE Transactions on Geoscience and Remote Sensing*, *48*(6), 2603-2612.

Dadon, A., Ben-Dor, E., Beyth, M., & Karnieli, A. (2011). Examination of spaceborne imaging spectroscopy data utility for stratigraphic and lithologic mapping. *Journal of Applied Remote Sensing*, *5*(1), 053507-1-053507-14.

Digital imaging and remote sensing image generation (DIRSIG). (2016). Retrieved October 2, 2016, from https://dirsig.cis.rit.edu/

El Gamal, A., & Eltoukhy, H. (2005). CMOS image sensors. *IEEE Circuits and Devices Magazine*, *21*(3), 6-20.

Feingersh, T., & Dor, E. B. (2015). SHALOM – A commercial hyperspectral space mission. In S. Qian (Ed.), *Optical payloads for space missions* (pp. 247). John Wiley & Sons.

Gaffey, S. J. (1986). Spectral reflectance of carbonate minerals in the visible and near infrared (0.35-2.55 microns); calcite, aragonite, and dolomite. *American Mineralogist, 71*(1-2), 151-162.

Goetz, A. F., Vane, G., Solomon, J. E., & Rock, B. N. (1985). Imaging spectrometry for earth remote sensing. *Science, 228*(4704), 1147-1152.

Gomez, C., Delacourt, C., Allemand, P., Ledru, P., & Wackerle, R. (2005). Using ASTER remote sensing data set for geological mapping in Namibia. *Physics and Chemistry of the Earth, Parts A/B/C, 30*(1), 97-108.

Gong, P., & Howarth, P. (1990). An assessment of some factors influencing multispectral land-cover classification. *Photogrammetric Engineering and Remote Sensing, 56*(5), 597-603.

Green, A. A., Berman, M., Switzer, P., & Craig, M. D. (1988). A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE Transactions on Geoscience and Remote Sensing, 26*(1), 65-74.

Green, A. A., Craig, M. D., & Shi, C. (1988b). The application of the minimum noise fraction transform to the compression and cleaning of hyper-spectral remote sensing data. In *Geoscience and Remote Sensing Symposium, 1988. IGARSS'88. Remote Sensing: Moving Toward the 21st Century., International* (pp. 1807-1807). IEEE.

Guanter, L., Segl, K., & Kaufmann, H. (2009). Simulation of optical remote-sensing scenes with application to the EnMAP hyperspectral mission. *IEEE Transactions on Geoscience and Remote Sensing, 47*(7), 2340-2351.

Guanter, L., Kaufmann, H., Segl, K., Foerster, S., Rogass, C., Chabrillat, S., ... & Straif, C. (2015). The EnMAP spaceborne imaging spectroscopy mission for earth observation. *Remote Sensing, 7*(7), 8830-8857.

Guo, B., Gunn, S. R., Damper, R. I., & Nelson, J. D. (2006). Band selection for hyperspectral image classification using mutual information. *IEEE Geoscience and Remote Sensing Letters, 3*(4), 522-526.

Gupta, R. K., Prasad, T. S., Rao, P. K., & Manikavelu, P. B. (2000). Problems in upscaling of high resolution remote sensing data to coarse spatial resolution over land surface. *Advances in Space Research, 26*(7), 1111-1121.

Harris, J. R. (2007). Remote Predictive Mapping (RPM): A not so new paradigm for mapping Canada's North. *Geoscience Canada, 34*(3). 91-92

Harris, J. R. (2008). *Remote Predictive Mapping: An Aid for Northern Mapping* (Open File 5643), Geological Survey of Canada.

Harris, J. R., He, J., Rainbird, R., & Behnia, P. (2014). Remote predictive mapping 6. A comparison of different remotely sensed data for classifying bedrock types in

Canada's Arctic: Application of the robust classification method and random forests. *Geoscience Canada, 41*(4)

Harris, J. R., Ponomarev, P., Shang, J., & Rogge, D. (2006). Noise reduction and best band selection techniques for improving classification results using hyperspectral data: Application to lithological mapping in canada's arctic. *Canadian Journal of Remote Sensing, 32*(5), 341-354.

Harris, J. R., Ponomarev, P., Shang, S., Budkewitsch, P., & Rogge, D. (2006). *A comparison of automatic and supervised methods for extracting lithological end members from hyperspectral data: Application to lithological mapping in southern Baffin Island, Nunavut* (Geological Survey of Canada, Current Research, 2006-C4).

Harris, J.R., Rencz, A., Ballantyne, B., & Sheridon, C. (1998). Mapping altered rocks using LANDSAT TM and lithogeochemical data: Sulphurets-Brucejack Lake district, British Columbia, Canada. *Photogrammetric Engineering and Remote Sensing, 64*(4), 309-322.

Harris, J. R., Rogge, D., Hitchcock, R., Ijewliw, O., & Wright, D. (2005). Mapping lithology in Canada's Arctic: Application of hyperspectral data using the minimum noise fraction transformation and matched filtering. *Canadian Journal of Earth Sciences, 42*(12), 2173-2193.

Hay, G. J., Niemann, K. O., & McLean, G. F. (1996). An object-specific image-texture analysis of H-resolution forest imagery. *Remote Sensing of Environment, 55*(2), 108-122.

Hobbs, P. C. (2011). *Building electro-optical systems: making it all work* (Vol. 71). New York, New York: John Wiley & Sons.

Holst, G. C. (1998). CCD arrays, cameras, and displays (Vol. 38). *SPIE-The International Society for Optical Engineering*, Bellingham, Washington: SPIE Optical Engineering Press.

Holst, G. C. (2009). Scientific CMOS image sensors. *Laser & Photonics, 5*, 18-21.

Horning, N. (2010). Random Forests: An algorithm for image classification and generation of continuous fields data sets. *Proceedings of the International Conference on Geoinformatics for Spatial Infrastructure Development in Earth and Allied Sciences (pp. 9-11)*, Osaka, Japan.

Horning, N. (2013). *Training guide for using Random Forests to classify satellite images - v9*. American Museum of Natural History, Center for Biodiversity and Conservation. Retrieved 2016, from http://biodiversityinformatics.amnh.org/.

Hsieh, P., Lee, L. C., & Chen, N. (2001). Effect of spatial resolution on classification errors of pure and mixed pixels in remote sensing. *IEEE Transactions on Geoscience and Remote Sensing, 39*(12), 2657-2663.

Hubbard, B. E., Crowley, J. K., & Zimbelman, D. R. (2003). Comparative alteration mineral mapping using visible to shortwave infrared (0.4-2.4/spl mu/m) Hyperion, ALI, and ASTER imagery. *IEEE Transactions on Geoscience and Remote Sensing, 41*(6), 1401-1410.

Hunt, G. R. (1977). Spectral signatures of particulate minerals in the visible and near infrared. *Geophysics, 42*(3), 501-513.

Hunt, G. R., & Ashley, R. P. (1979). Spectra of altered rocks in the visible and near infrared. *Economic Geology, 74*(7), 1613-1629.

Irons, J. R., Markham, B. L., Nelson, R. F., Toll, D. L., Williams, D. L., Latty, R. S., et al. (1985). The effects of spatial resolution on the classification of thematic mapper data. *International Journal of Remote Sensing, 6*(8), 1385-1403.

Jensen, J. R. (2007). *Remote Sensing of the environment: An Earth resource perspective* 2/e. Pearson Education: India.

Jensen, J. R., & Lulla, K. (1987). *Introductory digital image processing: A remote sensing perspective.* Upper Saddle River, NJ: Prentice Hall.

Jerram, P. A., Fryer, M., Pratlong, J., Pike, A., Walker, A., Dierickx, B., et al. (2010). Hyperspectral CMOS imager. *International Conference on Space Optics* (pp. 8), Rhodes, Greece.

Justice, C. O., Markham, B. L., Townshend, J. R. G., & Kennard, R. L. (1989). Spatial degradation of satellite data. *International Journal of Remote Sensing, 10*(9), 1539-1561.

Kalinowski, A., & Oliver, S. (2004). *ASTER mineral index processing manual* (Internal Report 36*).* Australia: Geoscience Australia.

Kasap, S., Ruda, H., & Boucher, Y. (2009). *Cambridge illustrated handbook of optoelectronics and photonics*. New York: Cambridge University Press.

Kaufmann, H., Chabrillat, S., Segl, K., Rogass, C., Hofer, S., Müller, A., et al. (2011). *The potential of imaging spectroscopy for geological mapping and mineral exploration [PDF document]*. GRSG Workshop, December 6-9, 2011, ESRIN, Frascati, Italy: Retrieved from http://earth.esa.int/workshops/grsg2011/files/1_kaufmann.pdf

Kavzoglu, T. (2004). Simulating Landsat ETM+ imagery using DAIS 7915 hyperspectral scanner data. *International Journal of Remote Sensing*, 25(22), 5049-5067.

Kerekes, J. P., & Baum, J. E. (2002). Spectral imaging system analytical model for subpixel object detection. *IEEE Transactions on Geoscience and Remote Sensing*, *40*(5), 1088-1101.

Kerekes, J. P., & Baum, J. E. (2003). Hyperspectral imaging system modeling. *Lincoln Laboratory Journal*, *14*(1), 117-130.

Kerekes, J. P., & Landgrebe, D. A. (1989). *Modeling, simulation, and analysis of optical remote sensing systems* (No. NASA-CR-186648), School of Electrical Engineering, Purdue University. Retrieved from https://engineering.purdue.edu/~landgreb/KerekesTR-EE89-49.pdf

Kruse, F. A, Boardman, J. W., Huntington, J. W., Mason, P., & Quigley, M. A. (2002). Evaluation and validation of EO-1 Hyperion for geologic mapping. *Geoscience and*

Remote Sensing Symposium, 2002 (IGARSS'02). 2002 IEEE International (pp. 593-595).

Kruse, F. A. (2000). The effects of spatial resolution, spectral resolution, and SNR on geologic mapping using hyperspectral data, northern Grapevine Mountains, Nevada. *Proceedings of the 9th JPL Airborne Earth Science Workshop* (Vol. 1, No. 9), Jet Propulsion Laboratory Publication.

Kruse, F. A., Boardman, J. W., & Huntington, J. F. (2003). Comparison of airborne hyperspectral data and EO-1 Hyperion for mineral mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 41(6), 1388-1400.

Kruse, F. A., & Perry, S. L. (2013). Mineral mapping using simulated WorldView-3 short-wave-infrared imagery. *Remote Sensing, 5*(6), 2688-2703.

Kruse, F. A., Taranik, J. V., Coolbaugh, M., Michaels, J., Littlefield, E. F., Calvin, W. M., & Martini, B. A. (2011). Effect of reduced spatial resolution on mineral mapping using imaging spectrometry—examples using Hyperspectral Infrared Imager (HyspIRI)-simulated data. *Remote Sensing*, 3(8), 1584-1602.

Kuosmanen, V., Laitinen, J., & Arkimaa, H. (2005). A comparison of hyperspectral airborne HyMap and spaceborne Hyperion data as tools for studying the environmental impact of talc mining in Lahnaslampi, NE Finland. *4th EARSeL Workshop on Imaging Spectroscopy. New Quality in Environmental Studies* (pp. 397-407), Warsaw, Poland.

Lee, C. M., Cable, M. L., Hook, S. J., Green, R. O., Ustin, S. L., Mandl, D. J., et al. (2015). An introduction to the NASA hyperspectral InfraRed imager (HyspIRI) mission and preparatory activities. *Remote Sensing of Environment, 167*, 6-19.

Legleiter, C., Marcus, W. A., & Lawrence, R. (2002). Effects of sensor resolution on mapping instream habitats. *Photogrammetric Engineering and Remote Sensing*, 68(8), 801-807.

Leverington, D. W. (2010). Discrimination of sedimentary lithologies using Hyperion and Landsat Thematic Mapper data: a case study at Melville Island, Canadian High Arctic. *International Journal of Remote Sensing*, *31*(1), 233-260.

Liao, L. B., Jarecke, P. J., Gleichauf, D. A., & Hedman, T. R. (2000). Performance characterization of the Hyperion imaging spectrometer instrument. *International Symposium on Optical Science and Technology* (pp. 264-275).

Lin, H., Wang, J., Bo, Y., & Bian, J. (2008). The Impacts of Landscape Patterns on the Accuracy of Remotely Sensed Data Classification. *Proceedings of the 8th International Symposium on Special Accuracy Assessment in Natural Resources and Environmental Sciences* (pp. 219-225), Shanghai, China.

Liu, B., Zhang, L., Zhang, X., Zhang, B., & Tong, Q. (2009). Simulation of EO-1 Hyperion data from ALI multispectral data based on the spectral reconstruction approach. *Sensors*, *9*(4), 3090-3108.

Lopinto, E., & Ananasso, C. (2013). The Prisma hyperspectral mission. *Proceedings of the 33rd EARSeL Symposium Towards Horizon 2020: Earth Observation and Social Perspectives* (pp. 135-146*)*, Matera, Italy.

Lorenz, H. (2004). Integration of Corona and Landsat Thematic Mapper data for bedrock geological studies in the high Arctic. *International Journal of Remote Sensing*, 25(22), 5143-5162.

Lu, D., & Weng, Q. (2007). A survey of image classification methods and techniques for improving classification performance. *International Journal of Remote Sensing*, 28(5), 823-870.

Lyon, R. J. P. (1965). Analysis of rocks by spectral infrared emission (8 to 25 microns). *Economic Geology, 60*(4), 715-736.

Markham, B. L. (1985). The Landsat sensors' spatial responses. *IEEE Transactions on Geoscience and Remote Sensing*, (6), 864-875.

Matsunaga, T., Iwasaki, A., Tsuchida, S., Tanii, J., Kashimura, O., Nakamura, R., et al. (2013). Current status of hyperspectral imager suite (HISUI). *Geoscience and Remote Sensing Symposium (IGARSS), 2013 IEEE International* (pp. 3510-3513).

Matsunaga, T., Yamamoto, S., Kato, S., Kashimura, O., Tachikawa, T., Ogawa, K., et al. (2011). Simulation of operation of future Japanese spaceborne hyperspectral imager: HISUI. *SPIE Remote Sensing* (pp. 818109-1-818109-8).

Meola, J., Eismann, M. T., Moses, R. L., & Ash, J. N. (2011). Modeling and estimation of signal-dependent noise in hyperspectral imagery. *Applied Optics, 50*(21), 3829-3846.

Michel, S., Gamet, P., & Lefevre-Fonollosa, M. (2011). HYPXIM—A hyperspectral satellite defined for science, security and defence users. *Third Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS)*, (pp. 1-4).

Ming, D., Luo, J., Li, L., & Song, Z. (2010). Modified local variance based method for selecting the optimal spatial resolution of remote sensing image. *2010 18th International Conference on Geoinformatics (*pp. 1-6).

Mouroulis, P. Z. (1999). Spectral and spatial uniformity in pushbroom imaging spectrometers. *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation (*pp. 133-141).

Neville, R. A., Levesque, J., Staenz, K., Nadeau, C., Hauff, P., & Borstad, G. A. (2003). Spectral unmixing of hyperspectral imagery for mineral exploration: comparison of results from SFSI and AVIRIS. *Canadian Journal of Remote Sensing,* 29(1), 99-110.

Nieke, J., Solbrig, M., & Neumann, A. (1999). Noise contributions for imaging spectrometers. *Applied Optics, 38*(24), 5191-5194.

Niemann, K. O., Goodenough, D. G., & Hay, G. J. (1997). Effect of scale on the information content in remote sensing imagery. *Geoscience and Remote Sensing, 1997. IGARSS'97. Remote Sensing-A Scientific Vision for Sustainable Development, 1997* (pp. 664-666).

Noerdlinger, P. D. (1999). Atmospheric refraction effects in Earth remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing, 54*(5), 360-373.

Oppelt, N., & Mauser, W. (2007). The airborne visible/infrared imaging spectrometer AVIS: Design, characterization and calibration. *Sensors, 7*(9), 1934-1953.

Otterman, J., & Fraser, R. S. (1979). Adjacency effects on imaging by surface reflection and atmospheric scattering: cross radiance to zenith. *Applied Optics, 18*(16), 2852-2860.

Pearlman, J. S., Barry, P. S., Segal, C. C., Shepanski, J., Beiso, D., & Carman, S. L. (2003). Hyperion, a space-based imaging spectrometer, *IEEE Transactions on Geoscience and Remote Sensing, 41*(6), 1160-1173.

Pearlman, J., Carman, S., Segal, C., Jarecke, P., Clancy, P., & Browne, W. (2001). Overview of the Hyperion imaging spectrometer for the NASA EO-1 mission. *Geoscience and Remote Sensing Symposium, 2001. IGARSS'01. IEEE 2001 International* (pp. 3036-3038). IEEE.

Peddle, D. R., Boulton, R. B., Pilger, N., Bergeron, M., & Hollinger, A. (2008). Hyperspectral detection of chemical vegetation stress: evaluation for the Canadian HERO satellite mission. *Canadian Journal of Remote Sensing, 34*(sup1), S198-S216.

Polder, G., Van Der Heijden, G. W. A. M., Keizer, L. P., & Young, I. T. (2003). Calibration and characterisation of imaging spectrographs. *Journal of Near Infrared Spectroscopy, 11*(3), 193-210.

Praks, J., Kestila, A., Hallikainen, M., Saari, H., Antila, J., Janhunen, P., & Vainio, R. (2011). Aalto-1 - An experimental nanosatellite for hyperspectral remote sensing. *Geoscience and Remote Sensing Symposium, 2011(IGARSS)* (pp. 4367-4370).

Qian, S. E. (2016). *Optical Payloads for Space missions*. West Sussex, UK: John Wiley & Sons.

Rahman, A. F., Gamon, J. A., Sims, D. A., & Schmidts, M. (2003). Optimum pixel size for hyperspectral studies of ecosystem function in southern California chaparral and grassland. *Remote Sensing of Environment, 84*(2), 192-207.

Rajesh, H. M. (2004). Application of remote sensing and GIS in mineral resource mapping - An overview. *Journal of Mineralogical and Petrological Sciences, 99*(3), 83-103.

Resmini, R. G., Kappus, M. E., Aldrich, W. S., Harsanyi, J. C., & Anderson, M. (1997). Mineral mapping with hyperspectral digital imagery collection experiment (HYDICE) sensor data at Cuprite, Nevada, USA. *International Journal of Remote Sensing, 18*(7), 1553-1570.

Richards, J. A., & Richards, J. A. (1999). *Remote sensing digital image analysis* (Vol. 3). Berlin, Germany: Springer.

Richter, R., Bachmann, M., Dorigo, W., & Muller, A. (2006). Influence of the adjacency effect on ground reflectance measurements. *Geoscience and Remote Sensing Letters, IEEE, 3*(4), 565-569.

Richter, R., Kellenberger, T., & Kaufmann, H. (2009). Comparison of topographic correction methods. *Remote Sensing, 1*(3), 184-196.

Rivard, B., Rogge, D., Jeng, J., Grant, B., & Pardy, J. (2010). Hyperspectral sensing in support of mineral exploration in the Arctic: Example from the Cape Smith nickel belt, Canada. *Proceedings of the 'Hyperspectral 2010 Workshop'*, Frascati, Italy.

Rockwell, B. W., & Hofstra, A. H. (2008). Identification of quartz and carbonate minerals across northern Nevada using ASTER thermal infrared emissivity data—Implications for geologic mapping and mineral resource investigations in well-studied and frontier areas. *Geosphere, 4*(1), 218-246.

Rogalski, A. (2012). Progress in focal plane array technologies. *Progress in Quantum Electronics, 36*(2), 342-473.

Rogge, D., Rivard, B., Segl, K., Grant, B., & Feng, J. (2014). Mapping of NiCu–PGE ore hosting ultramafic rocks using airborne and simulated EnMAP hyperspectral imagery, Nunavik, Canada. *Remote Sensing of Environment, 152*, 302-317.

Rowan, L. C., Hook, S. J., Abrams, M. J., & Mars, J. C. (2003). Mapping hydrothermally altered rocks at Cuprite, Nevada, using the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER), a new satellite-imaging system. *Economic Geology, 98*(5), 1019-1027.

Rowan, L. C., & Mars, J. C. (2003). Lithologic mapping in the Mountain Pass, California area using advanced spaceborne thermal emission and reflection radiometer (ASTER) data. *Remote Sensing of Environment, 84*(3), 350-366.

San, B. T. & Suzen, M.L. (2010). Evaluation of different atmospheric correction algorithms for EO-1 Hyperion imagery. International Archives of the Photogrammetry, *Remote Sensing and Spatial Information Science*, Volume XXXVIII, Part 8, Kyoto, Japan.

Schetselaar, E. M., & Ryan, J. J. (2009). Remote predictive mapping of the Boothia mainland area, Nunavut, Canada: an iterative approach using Landsat ETM, aeromagnetic, and geological field data. *Canadian Journal of Remote Sensing, 35* (sup1), S72-S94.

Schetselaar, E., & de Kemp, E. A. (2000). Image classification from Landsat TM, airborne magnetics and DEM data for mapping Paleoproterozoic bedrock units, Baffin Island, Nunavut, Canada. *International Archives of Photogrammetry and Remote Sensing*, *33*(B7/3; PART 7), 1325-1332.

Schlapfer, D., Nieke, J., & Itten, K. I. (2007). Spatial PSF nonuniformity effects in airborne pushbroom imaging spectrometry data. *IEEE Transactions on Geoscience and Remote Sensing, 45*(2), 458-468.

Schwind, P., Müller, R., Palubinskas, G., & Storch, T. (2012). An in-depth simulation of EnMAP acquisition geometry. *ISPRS Journal of Photogrammetry and Remote Sensing, 70*, 99-106.

Segl, K., Guanter, L., & Kaufmann, H. (2009). Forward simulation of hyperspectral remote sensing images in the frame of the EnMAP mission. *Imaging Spectroscopy: Innovative Tool for Scientific and Commercial Environmental Applications, 6th EarSEL SIG IS Workshop* (pp. 16-19). Tel Aviv, Israel

Segl, K., Guanter, L., Rogass, C., Kuester, T., Roessner, S., Kaufmann, H., ... & Hofer, S. (2012). EeteS—The EnMAP end-to-end simulation tool. *IEEE Journal of,Selected Topics in Applied Earth Observations and Remote Sensing*, *5*(2), 522-530.

Shaw, G. E. (1995). The Arctic haze phenomenon. *Bulletin of the American Meteorological Society*, *76*(12), 2403-2413.

Sherlock, R. L. & Carpenter, R. L. (2003). *Bedrock geology of the Wolverine-Doris corridor, Hope Bay volcanic belt, Nunavut* (Open File 1553), Geological Survey of Canada.

Shippert, P. (2003). Introduction to hyperspectral image analysis. *Online Journal of Space Communication, 3*.

Smirnoff, A., Huot-Vézina, G., Paradis, S. J., & Boivin, R. (2012). Generalizing geological maps with the GeoScaler software: The case study approach. *Computers & Geosciences, 40*, 66-86.

Smith, J. H., Stehman, S. V., Wickham, J. D., & Yang, L. (2003). Effects of landscape characteristics on land-cover class accuracy. *Remote Sensing of Environment, 84*(3), 342-349.

Smith, J. H., Wickham, J. D., Stehman, S. V., & Yang, L. (2002). Impacts of patch size and land-cover heterogeneity on thematic image classification accuracy. *Photogrammetric Engineering and Remote Sensing*, *68*(1), 65-70.

Smith, S. W. (1997). *The Scientist and Engineer's Guide to Digital Sound Processing*. Retrieved from http://www.dspguide.com/pdfbook.htm/

Sola, I., Gonzalez-Audicana, M., Alvarez-Mozos, J., & Torres, J. L. (2014). Synthetic images for evaluating topographic correction algorithms. *IEEE Transactions on Geoscience and Remote Sensing*, *52*(3), 1799-1810.

Staenz, K. (1992). A decade of imaging spectrometry in Canada. *Canadian Journal of Remote Sensing, 18*(4), 187-197.

Staenz, K., Budkewitsch, P., Neville, R. A., Hitchcock, R., & Nadeau, C. (2001). Spectral unmixing of rock/mineral targets based on different spatial resolution hyperspectral data. *ISSSR'01: Proceedings of the International Symposium on Spectral Sensing Research* (pp. 10-15).

Staenz, K., Nadeau, C., Secker, J., & Budkewitsch, P. (2000). Spectral unmixing applied to vegetated environments in the Canadian Arctic for mineral mapping. *International Archives of Photogrammetry and Remote Sensing, 33*(B7/4; PART 7), 1464-1471.

Staenz, K., Neville, R. A., Hitchcock, R., Assouad, P., Omari, K., Sun, L., ... & White, H. P. (2005). The potential of the proposed Canadian HERO mission for geoscience applications. *Proceedings of the 4th EARSeL Workshop on Imaging Spectroscopy* (pp. 27-29).

Staenz, K., & Held, A. (2012). Summary of current and future terrestrial civilian hyperspectral spaceborne systems. *Geoscience and Remote Sensing Symposium (IGARSS), 2012* (pp. 123-126). IEEE.

Strobl, C., Boulesteix, A. L., Zeileis, A., & Hothorn, T. (2007). Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC Bioinformatics, 8*(1), 25

Swain, P. H., & King, R. C. (1973). *Two effective feature selection criteria for multispectral remote sensing*. LARS Technical Report Number 042673. Retrieved from http://docs.lib.purdue.edu/larstech/39/.

Swayze, G. A., Clark, R. N., Goetz, A. F., Chrien, T. G., & Gorelick, N. S. (2003). Effects of spectrometer band pass, sampling, and signal-to-noise ratio on spectral

identification using the Tetracorder algorithm. *Journal of Geophysical Research: Planets*, *108*(E9).

Swayze, G. A., Clark, R. N., Kruse, F. A., Sutley, S., & Gallagher, A. (1992). Ground-truthing AVIRIS mineral mapping at Cuprite, Nevada. *JPL, Summaries of the Third Annual JPL Airborne Geoscience Workshop*. (Vol. 1) (pp. 47-49).

Tack, N., Lambrechts, A., Soussan, P., & Haspeslagh, L. (2012). A compact, high-speed, and low-cost hyperspectral imager. *Proceedings of SPIE OPTO, Silicon Photonics VII* (pp. 82660Q-1-82660Q-12), International Society for Optics and Photonics.

Taylor, A.E.F. (2000). *Illumination fundamentals*. Retrieved from http://optics.synopsys.com/lighttools/pdfs/illuminationfund.pdf

Theuwissen, A. J. (2008). CMOS image sensors: State-of-the-art. *Solid-State Electronics*, *52*(9), 1401-1406.

Tobler, W.R. (1987). Measuring Spatial Resolution. *Proceedings of the Land Resources Information Systems Conference* (pp.12–16), Beijing, China.

Tobler, W. R. (1989). Frame independent spatial analysis. *Accuracy of spatial databases*, 115-122.

Townshend, J. R. (1981). The spatial resolving power of earth resources satellites. *Progress in Physical Geography*, *5*(1), 32-55.

Turner, M. G., O'Neill, R. V., Gardner, R. H., & Milne, B. T. (1989). Effects of changing spatial scale on the analysis of landscape pattern. *Landscape ecology*, *3*(3-4), 153-162.

van der Meer, F., van der Werff, H., & de Jong, S. M. (2009). Pre-processing of optical imagery. In T.A. Warner (Ed.), *The SAGE Handbook of Remote Sensing*. London, UK: SAGE Publishers.

van der Meer, F. D., van der Werff, H., van Ruitenbeek, F. J., Hecker, C. A., Bakker, W. H., Noomen, M. F., ... & Woldai, T. (2012). Multi-and hyperspectral geologic remote sensing: A review. *International Journal of Applied Earth Observation and Geoinformation, 14*(1), 112-128.

Villafranca, A. G., Corbera, J., Martín, F., & Marchán, J. F. (2012). Limitations of hyperspectral earth observation on small satellites. *Journal of Small Satellites, 1*(1), 19-29.

Wang, D., Zhang, T., & Kuang, H. (2012). Relationship between the charge-coupled device signal-to-noise ratio and dynamic range with respect to the analog gain. *Applied Optics*, 51(29), 7103-7114.

White, H. P., Sun, L., Gauthier, R., Budkewitsch, P., Guanter, L., Segl, K., & Kaufmann, H. (2010) Independent evaluation of Enmap sensor for the geological mapping, *Proceedings of the 'Hyperspectral 2010 Workshop'*, Frascati, Italy.

Wickert, L. M., & Budkewitsch, P. (2004). ASTER - a geological mapping tool for Canada's North. Case Study: The Belcher Islands, Hudson Bay, Nunavut, Canada. Geoscience and Remote Sensing Symposium, 2004. IGARSS'04. Proceedings. 2004 IEEE International (Vol. 2, pp. 1300-1303). IEEE.

Wiens, J. A. (1989). Spatial scaling in ecology. *Functional ecology, 3*(4), 385-397.

Woodcock, C. E., & Strahler, A. H. (1987). The factor of scale in remote sensing. *Remote Sensing of Environment, 21*(3), 311-332.

Woodcock, C. E., Strahler, A. H., & Jupp, D. L. (1988). The use of variograms in remote sensing: I. Scene models and simulated images. *Remote Sensing of Environment, 25*(3), 323-348.

Wulder, M. A., White, J. C., Niemann, K. O., & Nelson, T. (2004). Comparison of airborne and satellite high spatial resolution data for the identification of individual

trees with local maxima filtering. *International Journal of Remote Sensing, 25*(11), 2225-2232.

Yokoya, N., Miyamura, N., & Iwasaki, A. (2010, October). Preprocessing of hyperspectral imagery with consideration of smile and keystone properties. *SPIE Asia- Pacific Remote Sensing* (pp. 78570B-78570B). International Society for Optics and Photonics.

Zhang, J., Chen, J., Zou, B., & Zhang, Y. (2012). Modeling and simulation of polarimetric hyperspectral imaging process. *IEEE Transactions on Geoscience and Remote Sensing, 50*(6), 2238-2253.

Zhang, J., Rivard, B., & Sánchez-Azofeifa, A. (2005). Spectral unmixing of normalized reflectance data for the deconvolution of lichen and rock mixtures. *Remote Sensing of Environment, 95*(1), 57-66.

Zhang, J., Zhang, X., Zou, B., & Chen, D. (2010). On hyperspectral image simulation of a complex woodland area. *IEEE Transactions on Geoscience and Remote Sensing, 48*(11), 3889-3902.

Zhang, X., & Pazner, M. (2007). Comparison of lithologic mapping with ASTER, Hyperion, and ETM data in the southeastern Chocolate Mountains, USA. *Photogrammetric Engineering & Remote Sensing, 73*(5), 555-561.

Zhao, H., Jia, G., & Li, N. (2010). Transformation from hyperspectral radiance data to data of other sensors based on spectral superresolution. *IEEE Transactions on Geoscience and Remote Sensing*, *48*(11), 3903-3912.

Appendix A: Acquisition Table of Flight Dates and Times

Flight Line	Date	Time (UTC)
1	August 21, 2009	18:27:30
2	August 21, 2009	18:36:20
3	August 21, 2009	18:45:22
4	August 21, 2009	18:00:00
5	August 21, 2009	19:02:22
6	August 21, 2009	19:10:48
7	August 21, 2009	19:19:38
8	August 21, 2009	19:28:18
9	August 21, 2009	19:36:49
10	August 21, 2009	19:45:30
11	August 21, 2009	19:23:00
12	August 23, 2009	19:35:56
13	August 23, 2009	19:46:31

Appendix B: CRAN R Scripts for Sensor Simulation

Five CRAN R scripts make up the sensor simulation modelling and are included in this appendix: 1) defining the user variables and calling each of the necessary following scripts, 2) loading the necessary R packages and libraries, 3) spatial convolution, 4) spectral convolution, and 5) noise modelling.

```
#########################################################################################################################
# 0. Simulation.R
# R script for convolving hyperspectral data. The user can spatially and/or spectrally convolve imagery as well as add
# signal and signal independent noise.
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada - Natural Resources Canada
# Script completed December, 2016
#########################################################################################################################
#Remove all current objects to clear memory
rm(list = ls())
gc()

#start a stopwatch to time the script
starttime0 <- proc.time()

# Set the working directory300
setwd("U:/Thesis/Data/TestData6") # *

# Print warnings
warnings()

#####################################
# Load required packages and libraries:
#####################################
print ("Loading required packages and libraries")
#source("U:/Thesis/R scripts/1. Loading packages.R")
print ("Loaded packages and libraries")

# set the temporary directory during processing to a location with sufficient space
options(rasterTmpDir='U:/temp/') # *
# set the default output file type for rasters to GeoTiff
rasterOptions(format="GTiff")
#####################################
# Spatial convolution parameters:
#####################################
# Set the input dataset and header file
rBrickInput <- brick("U:/Thesis/Data/UVICData/Window2.dat", package="raster") #modify value as necessary *


# set the resampling method. Accepable values are "Gaussian", "Aggregated", "Sinc", "Sinc Hamming", and "Sinc Blackman"
resamplingMethod = "Gaussian" # *

# Vector containing the resolution of the input brick
rres <- res (rBrickInput)

# Desired output resolution # *
desiredOutRes <- 30
rres <- tail (rres,1)
CellSizeDesRatio <- desiredOutRes/rres

if (CellSizeDesRatio != 15) {
  if (CellSizeDesRatio != 10) {break}
} {break} # stop the script if the ratio of input  to output spatial resolution cannot be accomplished by this script

# set the Gaussian blur scale. 2 = gaussian distribution. > 2 = more blur and <2 equals decreased blur. 0 creates no
# blur. This is in effect increasing or decreasing the height of the guassian distribution.
GaussianBlurScale = 0 # *

# write the projection information
coordinatesInput <- "+proj=utm +zone=10 +datum=NAD83 +units=m +no_defs + ellps=GRS80 +towgs84=0,0,0" # * Enter the
# projection information of the input file

#####################################
# Spectral convolution parameters:
#####################################
# manually enter wavelength band positions of the input imagery *
wavelength = c(408.790009,  410.920013,  413.040009,  415.170013,  417.299988,  419.420013,
               421.549988,  423.679993,  425.809998,  427.929993,  430.059998,  432.190002,
               434.309998,  436.470001,  438.690002,  440.920013,  443.140015,  445.369995,
               447.600006,  449.820007,  452.049988,  454.269989,  456.500000,  458.720001,
               460.950012,  463.179993,  465.399994,  467.630005,  469.850006,  472.079987,
               474.299988,  476.529999,  478.750000,  480.980011,  483.209991,  485.429993,
               487.660004,  489.880005,  492.109985,  494.329987,  496.559998,  498.779999,
               501.010010,  503.239990,  505.459991,  507.690002,  509.910004,  512.140015,
               514.359985,  516.590027,  518.809998,  521.039978,  523.270020,  525.489990,
               527.719971,  529.940002,  532.169983,  534.390015,  536.619995,  538.849976,
               541.070007,  543.299988,  545.520020,  547.809998,  550.119995,  552.429993,
               554.729980,  557.039978,  559.349976,  561.659973,  563.969971,  566.280029,
               568.580017,  570.890015,  573.200012,  575.510010,  577.820007,  580.130005,
               582.429993,  584.739990,  587.049988,  589.359985,  591.669983,  593.979980,
```

596.280029, 598.590027, 600.900024, 603.210022, 605.520020, 607.830017,
610.130005, 612.440002, 614.750000, 617.059998, 619.369995, 621.679993,
623.979980, 626.289978, 628.599976, 630.909973, 633.219971, 635.520020,
637.830017, 640.140015, 642.450012, 644.760010, 647.070007, 649.380005,
651.690002, 654.000000, 656.309998, 658.619995, 660.929993, 663.239990,
665.549988, 667.859985, 670.169983, 672.489990, 674.799988, 677.109985,
679.419983, 681.729980, 684.039978, 686.349976, 688.659973, 690.969971,
693.280029, 695.590027, 697.900024, 700.229980, 702.549988, 704.869995,
707.190002, 709.510010, 711.830017, 714.150024, 716.469971, 718.789978,
721.109985, 723.440002, 725.760010, 728.080017, 730.400024, 732.719971,
735.039978, 737.359985, 739.679993, 742.000000, 744.320007, 746.650024,
748.969971, 751.299988, 753.669983, 756.030029, 758.400024, 760.760010,
763.130005, 765.489990, 767.859985, 770.219971, 772.590027, 774.950012,
777.309998, 779.679993, 782.039978, 784.409973, 786.770020, 789.140015,
791.500000, 793.869995, 796.229980, 798.599976, 800.960022, 803.320007,
805.690002, 808.049988, 810.419983, 812.780029, 815.150024, 817.510010,
819.880005, 822.239990, 824.609985, 826.969971, 829.330017, 831.690002,
834.049988, 836.409973, 838.770020, 841.130005, 843.489990, 845.849976,
848.219971, 850.580017, 852.940002, 855.299988, 857.659973, 860.020020,
862.380005, 864.739990, 867.099976, 869.460022, 871.820007, 874.179993,
876.539978, 878.900024, 881.260010, 883.630005, 885.989990, 888.349976,
890.710022, 893.070007, 895.429993, 897.789978, 900.150024, 902.510010,
904.869995, 907.229980, 909.590027, 911.950012, 914.320007, 916.679993,
919.049988, 921.419983, 923.780029, 926.150024, 928.520020, 930.880005,
933.250000, 935.609985, 937.979980, 940.349976, 942.710022, 945.080017,
947.440002, 949.809998, 952.179993, 954.539978, 956.909973, 959.280029,
961.640015, 964.010010, 966.369995, 968.739990, 971.109985, 973.469971,
975.840027, 978.210022, 980.570007, 982.940002, 985.299988, 987.669983,
997.380005, 1003.650024, 1009.909973, 1016.179993, 1022.440002, 1028.709961,
1034.969971, 1041.229980, 1047.500000, 1053.760010, 1060.030029, 1066.290039,
1072.560059, 1078.819946, 1085.089966, 1091.349976, 1097.609985, 1103.880005,
1110.140015, 1116.410034, 1122.670044, 1128.939941, 1135.199951, 1141.469971,
1147.729980, 1153.989990, 1160.260010, 1166.520020, 1172.790039, 1179.050049,
1185.319946, 1191.579956, 1197.839966, 1204.109985, 1210.369995, 1216.640015,
1222.900024, 1229.170044, 1235.430054, 1241.699951, 1247.959961, 1254.219971,
1260.489990, 1266.750000, 1273.020020, 1279.280029, 1285.550049, 1291.810059,
1298.069946, 1304.339966, 1310.599976, 1316.869995, 1323.130005, 1329.400024,
1335.660034, 1341.930054, 1348.189941, 1354.449951, 1360.719971, 1366.979980,
1373.250000, 1379.510010, 1385.780029, 1392.040039, 1398.310059, 1404.569946,
1410.829956, 1417.099976, 1423.359985, 1429.630005, 1435.890015, 1442.160034,
1448.420044, 1454.680054, 1460.949951, 1467.209961, 1473.479980, 1479.739990,
1486.010010, 1492.270020, 1498.540039, 1504.800049, 1511.060059, 1517.329956,
1523.589966, 1529.859985, 1536.119995, 1542.390015, 1548.650024, 1554.910034,
1561.180054, 1567.439941, 1573.709961, 1579.969971, 1586.239990, 1592.500000,
1598.770020, 1605.030029, 1611.290039, 1617.560059, 1623.819946, 1630.089966,
1636.349976, 1642.619995, 1648.880005, 1655.150024, 1661.410034, 1667.670044,
1673.939941, 1680.199951, 1686.469971, 1692.729980, 1699.000000, 1705.260010,
1711.520020, 1717.790039, 1724.050049, 1730.319946, 1736.579956, 1742.849976,
1749.109985, 1755.380005, 1761.640015, 1767.900024, 1774.170044, 1780.430054,
1786.699951, 1792.959961, 1799.229980, 1805.489990, 1811.760010, 1818.020020,
1824.280029, 1830.550049, 1836.810059, 1843.079956, 1849.339966, 1855.609985,
1861.869995, 1868.130005, 1874.400024, 1880.660034, 1886.930054, 1893.189941,
1899.459961, 1905.719971, 1911.989990, 1918.250000, 1924.510010, 1930.780029,
1937.040039, 1943.310059, 1949.569946, 1955.839966, 1962.099976, 1968.359985,
1974.630005, 1980.890015, 1987.160034, 1993.420044, 1999.689941, 2005.949951,
2012.219971, 2018.479980, 2024.739990, 2031.010010, 2037.270020, 2043.540039,
2049.800049, 2056.070068, 2062.330078, 2068.600098, 2074.860107, 2081.120117,
2087.389893, 2093.649902, 2099.919922, 2106.179932, 2112.449951, 2118.709961,
2124.969971, 2131.239990, 2137.500000, 2143.770020, 2150.030029, 2156.300049,
2162.560059, 2168.830078, 2175.090088, 2181.350098, 2187.620117, 2193.879883,
2200.149902, 2206.409912, 2212.679932, 2218.939941, 2225.199951, 2231.469971,
2237.729980, 2244.000000, 2250.260010, 2256.530029, 2262.790039, 2269.060059,
2275.320068, 2281.580078, 2287.850098, 2294.110107, 2300.379883, 2306.639893,
2312.909912, 2319.169922, 2325.439941, 2331.699951, 2337.959961, 2344.229980,
2350.489990, 2356.760010, 2363.020020, 2369.290039, 2375.550049, 2381.810059,
2388.080078, 2394.340088, 2400.610107, 2406.870117, 2413.139893, 2419.399902,
2425.669922, 2431.929932, 2438.189941, 2444.459961, 2450.719971, 2456.989990,
2463.250000, 2469.520020, 2475.780029, 2482.040039, 2488.310059, 2494.570068)

# manually enter if wavelengths in the header file of the input data are stored as um or nm.
spectralResolution = "nm" # #modify value as necessary. Enter either "nm" or "um" only *

# Enter the Spectral Range of the VNIR sensor array of the desired output. 400 and 1000 for EnMap
VNIRminimumWavelength = 420 #modify value as necessary. #420 for EnMap *
VNIRmaximumWavelength = 1000 #modify value as necessary.#1000 for EnMap*

# Enter the Spectral Range of the SWIR sensor array of the desired output. 900 and 2450 for EnMap
SWIRminimumWavelength = 900 #modify value as necessary. #900 for EnMap *
SWIRmaximumWavelength = 2450 #modify value as necessary. #2450 for EnMap *

```
# Enter the desired spectral resolution for the VNIR bands
desiredSpectralResVNIR = 6.5 # has to be 6.5, 7.5 or 10. It is 6.5 for EnMap  *

# Enter the desired spectral resolution for the SWIR bands
desiredSpectralResSWIR = 10 # has to be 6.5, 7.5, or 10. It is 10 for EnMap *

#####################################
# Set Noise modelling parameters
#####################################
#Enter the desired SNR values.
desiredSNRVNIR <- 500 #* set the SNR estimate value for the VNIR. EnMap: 500
desiredSNRSWIR <- 150 #* set the SNR estimatevalue for the SWIR. EnMap: 150
desiredSNRVNIRWavelength <- 498 # * enter the band position of the SNR estimate for the VNIR value. EnMap: 498
desiredSNRSWIRWavelength <- 2200 # * enter the band position of the SNR estimate for the SWIR value EnMap: 2200

#Input Signal Independent noise parameters:
highMeasuredSNR = 1050 #the SNR at the highest measured reflectance *  1050 for EnMap *
highMeasuredSNRPosition = 550 #the band position of the SNR at the highest measured reflectance *  550 for EnMap *
noiseReflectance <- 3000 # * specify the amount of reflectance at the specified SNR parameters (30% for EnMap and
# entered as 3000)

#Define what the NA values are.
NAValues <- -1.7e+308 #Specify what value NAs values are stored as.*


#####################################
# Run spatial resampling script
#####################################
print ("Starting Spatial Resampling")
source("U:/Thesis/R scripts/2. Spatial Resample.R")
print ("Finished Running Spatial Resampling")

#####################################
# Run Spectral resampling script
#####################################
print ("Starting Spectral Resampling")
source("U:/Thesis/R scripts/3. Spectral Resample.R")
print ("Finished Running Spectral Resampling")

#####################################
# Run Spectral resampling script
#####################################
print ("Starting Noise modelling")
source("U:/Thesis/R scripts/4. Noise Modelling.R")
print ("Finished Running Noise modelling")

#stop the stopwatch to see how long the script took to run
proc.time() - starttime0
print (proc.time() - starttime0)
# end of script
```

```
#########################################################################################################################
# 1. Loading packages.R
# Script details:
# R script for loading necessary packages and libraries
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada (Natural Resources Canada)
# Script completed December, 2016
#########################################################################################################################
#Load necessary R packages
#Load hyperSpec package
if(require("hyperSpec")){
  print("hyperSpec is loaded correctly")
} else {
  print("trying to install hyperSpec")
  suppressWarnings(suppressMessages((install.packages("hyperSpec"))))
  if(require(hyperSpec)){
    print("hyperSpec installed and loaded")
  } else {
    stop("could not install hyperSpec")
  }
}

#Load rgl package
if(require("rgl")){
  print("rgl is loaded correctly")
} else {
  print("trying to install rgl")
  install.packages("rgl")
  if(require(rgl)){
    print("rgl installed and loaded")
  } else {
    stop("could not install rgl")
  }
}

#Load rgdal package
if(require("rgdal")){
  print("rgdal is loaded correctly")
} else {
  print("trying to install rgdal")
  install.packages("rgdal")
  if(require(rgdal)){
    print("rgdal installed and loaded")
  } else {
    stop("could not install rgdal")
  }
}

#Load maptools package
if(require("maptools")){
  print("maptools is loaded correctly")
} else {
  print("trying to install maptools")
  install.packages("maptools")
  if(require(maptools)){
    print("maptools installed and loaded")
  } else {
    stop("could not install maptools")
  }
}

#Load rgeos package
if(require("rgeos")){
  print("rgeos is loaded correctly")
} else {
  print("trying to install rgeos")
  install.packages("rgeos")
  if(require(rgeos)){
    print("rgeos installed and loaded")
  } else {
    stop("could not install rgeos")
  }
}

#Load ptw package
if(require("ptw")){
  print("ptw is loaded correctly")
} else {
  print("trying to install ptw")
  install.packages("ptw")
```

```
  if(require(ptw)){
    print("ptw installed and loaded")
  } else {
    stop("could not install ptw")
  }
}
library (ptw)


#Load raster package
if(require("raster")){
  print("raster is loaded correctly")
} else {
  print("trying to install raster")
  install.packages("raster")
  if(require(raster)){
    print("raster installed and loaded")
  } else {
    stop("could not install raster")
  }
}

#Load data.table package
if(require("data.table")){
  print("data.table is loaded correctly")
} else {
  print("trying to install data.table")
  #install.packages("data.table", repos="http://R-Forge.R-project.org")
  install.packages("data.table")
  if(require(data.table)){
    print("data.table installed and loaded")
  } else {
    stop("could not install data.table")
  }
}

#Load plot3D package
if(require("plot3D")){
  print("plot3D is loaded correctly")
} else {
  print("trying to install plot3D")
  install.packages("plot3D")
  if(require(plot3D)){
    print("plot3D installed and loaded")
  } else {
    stop("could not install plot3D")
  }
}

#list installed packages
ip <- as.data.frame(installed.packages()[,c(1,3:4)])
rownames(ip) <- NULL
ip <- ip[is.na(ip$Priority),1:2,drop=FALSE]
print(ip, row.names=FALSE)

# end of script
##########################################################################################
```

```
#####################################################################################################################
# 2. Spatial Resample.R
# Script details:
# R script for converting hyperspectral imagery from high resolution to lower resolution (default: 30 m) by resampling
# using a moving kernel simulating a nominal guassian, truncated sinc, sinc-hamming, sinc-Blackman function Point
# Spread Function (PSF).
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada - Natural Resources Canada
#
# Script completed December, 2016
#####################################################################################################################
#start a stopwatch to time the script
#starttime <- proc.time()

rBrickInputOriginal <- rBrickInput #copy the input data so that it retained before any changes

# Create a mirrored border around the dataset so that the resampling kernels have values
# to sample from to maintain it's extents.
# set a variable that is the buffer distance required by the windowed kernel based on the size required
if (CellSizeDesRatio == 10) {
  borderBuffer = 16}
if (CellSizeDesRatio == 15) {
  borderBuffer = 25 }
if (CellSizeDesRatio == 30) {
  borderBuffer = 46 }
if (CellSizeDesRatio == 6) {
  borderBuffer = 10}

#mirror the left side of the input dataset. THis is required as the windowed kernel will otherwise start inside the
# input dataset
xminextent = xmin(rBrickInput)
xmaxextent = xmin(rBrickInput) + (borderBuffer)
yminextent = ymin(rBrickInput)
ymaxextent = ymax(rBrickInput)
e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
fliprBrickInput <- crop(rBrickInput, e)
fliprBrickInput <- flip (fliprBrickInput, direction='x')

#mirror the right side
xminextent = xmax(rBrickInput) - (borderBuffer)
xmaxextent = xmax(rBrickInput)
e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
flipRasterBrick2 <- crop(rBrickInput, e)
flipRasterBrick2 <- flip (flipRasterBrick2, direction='x')
flipRasterBrick2 <- raster::shift(flipRasterBrick2, x=(borderBuffer*2), y=0) #the right side needs to be moved to able
# to add it to input data
shiftedInputRasterBrick <- raster::shift(rBrickInput, x=(borderBuffer), y=0) #the input data needs to be moved to the
# right to allow the mirrored data to be added
origin (shiftedInputRasterBrick) <- 0
origin (flipRasterBrick2) <- 0
origin (fliprBrickInput) <- 0
rBrickInput <- merge (shiftedInputRasterBrick, flipRasterBrick2, fliprBrickInput) #merge the mirrored areas

rm (fliprBrickInput)
rm (flipRasterBrick2)
#garbage collection
gc()

#now add mirrored values to the top
xminextent = xmin(rBrickInput)
xmaxextent = xmax(rBrickInput)
yminextent = ymax(rBrickInput) - (borderBuffer)
ymaxextent = ymax(rBrickInput)
e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
flipRasterBrick3 <- crop(rBrickInput, e)
flipRasterBrick3 <- flip (flipRasterBrick3, direction='y')
flipRasterBrick3 <- raster::shift(flipRasterBrick3, x=0, y=((borderBuffer)*2))

#now add mirrored values to the bottom
xminextent = xmin(rBrickInput)
xmaxextent = xmax(rBrickInput)
yminextent = ymin(rBrickInput)
ymaxextent = ymin(rBrickInput) + (borderBuffer)
e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
fliprBrickInputOriginal <- crop(rBrickInput, e)
fliprBrickInputOriginal <- flip (fliprBrickInputOriginal, direction='y')
#the top needs to be moved to able to add it to input data
shiftedInputRasterBrick <- raster::shift(rBrickInput, x=0, y=(borderBuffer)) #the input data needs to be moved upwards
# to allow the mirrored data to be added
origin (shiftedInputRasterBrick) <- 0
```

```
origin (flipRasterBrick3) <- 0
origin (fliprBrickInputOriginal) <- 0
rBrickInput <- merge (shiftedInputRasterBrick, flipRasterBrick3, fliprBrickInputOriginal) #merge the mirrored areas
rm (flipRasterBrick3)
rm (fliprBrickInputOriginal)
rm (shiftedInputRasterBrick)

# shift the raster back to its original coordinates
rBrickInput <- raster::shift(rBrickInput, x=-(borderBuffer), y=-(borderBuffer)) #the right side needs to be moved to
# able to add it to input data

#convert the convolved spatially raster brick to a GeoTiff file
outputfilename <- "input_Mirrored.tif"
#outputFile <- writeRaster(rBrickInput, filename = (outputfilename), overwrite = TRUE)
#garbage collection
gc()

#backup the raster brick before blurring
rasterBrickNoBlur <- rBrickInput
#############################################################################
# Gaussian blur
#Apply Guassian blur for simulating sensor jitter only if specified above. If a value was entered of 0 then skip
# this step.
if (GaussianBlurScale > 0){
  rasterBeforeBlur <- rBrickInput
  # Defining the 5x5 Gaussian Function spatial filter to simulate motion blur.
  y <- x <- seq(-2, 2, length=5) # length is changing the sampling interval
  standardDev = GaussianBlurScale/2.3548
  tempBrick1 <- subset(rBrickInput, 1) # temporarily subset the first band to create a raster brick as a start
  # now loop through each of the bands starting from the second band and writing to the first band sucessively.
  # For each band in the raster brick apply the blur to create a new raster brick and overwrite the input raster brick
  for (i in 1:nlayers(rBrickInput)){
    rband <- subset (rBrickInput, i)
    f <- function(x,y) {
      r <- sqrt(x^2+y^2); 1 * (1/(sqrt(2*pi*standardDev)))*(2.718281828459^((- ((r^2) / (2*(standardDev^2)))))) } #1 is
    # the vertical scale
    gaussianBlur <- outer(x, y, f)
    #Alter values of matrix so that they sum to 1 thus maintaining the range of values in the high res imagery
    gaussianBlur_Unity <- (gaussianBlur * 1 / sum(gaussianBlur))
    focal1 = focal(rband, w=gaussianBlur_Unity, na.rm=FALSE, pad=FALSE, NAonly=FALSE) #apply focal matrix to each band
    tempBrick1 <- addLayer(tempBrick1, focal1)
  }
  xminextent = xmin(rBrickInput)
  xmaxextent = xmax(rBrickInput)
  yminextent = ymin(rBrickInput)
  ymaxextent = ymax(rBrickInput)
  e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
  #tempBrick1 <- crop(tempBrick1, e)
  tempBrick1 <- dropLayer(tempBrick1,1) # drop the  first layer as it was temporary
  proj4string(tempBrick1) <- CRS(coordinatesInput)
  rBrickInput <- tempBrick1
}

#############################################################################################################################
# temporarily aggregate high res to a low res raster
highResR <- subset(rBrickInput,1)
lowResR <-aggregate(highResR, fact = (CellSizeDesRatio)) #aggregate to the lower resolution
#remove (highResR) # remove object as it's not needed any further
rm  (highResR)
#raster to point as a temporary step
aggPoints <- rasterToPoints(lowResR) #convert point to a raster
rm (lowResR)
#garbage collection
gc()
#############################################################################################################################
# If the user requested a Gaussian function PSF do the following
if (resamplingMethod == "Gaussian") {
  #Defining the PSF by a Gaussian Function...
  #Set the standard deviation as 4.2466. This value is determined by dividing the input/output image resolution ratio
  #by 2.3548 (a constant for FWHM).

  if (CellSizeDesRatio == 10) {
    y <- x <- seq(-15, 15, length = 31)
  } else if (CellSizeDesRatio == 30) {
    y <-
      x <-
      seq(-45, 45, length = 61) # when the ratio is 30, the length is changing the sampling interval
  } else if (CellSizeDesRatio == 15) {
    y <-
      x <-
```

```
    seq(-20, 20, length = 41) # when the ratio is 15, the length is changing the sampling interval
  }

  standardDev = CellSizeDesRatio / 2.3548

  f <- function(x,y) {
    r <- sqrt(x ^ 2 + y ^ 2); 1 * (1 / (sqrt(2 * pi * standardDev))) * (2.718281828459 ^((-((r ^ 2) / (2 * (standardDev ^ 2))))))
  } # 1 is the vertical scale

  psf_Gaussian <- outer(x, y, f)

  # copy the psf_Gaussian to maintain its's value before altering them in the next step.
  psf_Gaussian_norm = (psf_Gaussian - min(psf_Gaussian)) / (max(psf_Gaussian) -
                                                min(psf_Gaussian))

  #Alter values of matrix so that they sum to 1 thus maintaining the range of values in the high res imagery#
  psf_Gaussian = psf_Gaussian / sum(psf_Gaussian)
  psf_Gaussian_Unity <- (psf_Gaussian * 1 / sum(psf_Gaussian))
  #persp3D(x, y, psf_Gaussian_norm, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Gaussian", curtain = FALSE, border = "black", colkey = FALSE, zlab =
"PSF", ticktype = "detailed", zlim = c(-0.2, 1))

  ##########################################################################################################################
  #Apply Gaussian kernel matrix to imagery
  #create a loop selecting each of the bands and applying a focal matrix (kernel)

  tempBrick1 <- subset(rBrickInput, 1) # temporarily subset the first band to create a raster brick to as a starting point
  #tempBrick1 <- focal(tempBrick1, w = psf_Gaussian_Unity, na.rm = FALSE, pad = TRUE, padvalue = (borderBuffer - 1), NAonly = FALSE) #apply focal matrix to each
band
  #tempBrick1 <- focal(tempBrick1, w = psf_Gaussian_Unity, na.rm = FALSE, pad = FALSE, NAonly = FALSE) #apply focal matrix to each band

  aggPoints_gaus <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
  rasValue = extract(tempBrick1, aggPoints_gaus, method = 'simple')
  combinePointValue = cbind(aggPoints_gaus,rasValue)
  remove (rasValue)
  #garbage collection
  gc()
  tempBrick1 <- rasterFromXYZ(combinePointValue)

  # loop through each of the bands starting from the second band and writing to the first band sucessively
  for (i in 1:nlayers(rBrickInput)) {
    rband <- subset (rBrickInput, i)
    #focal1 = focal(rband, w = psf_Gaussian_Unity, na.rm = FALSE, padvalue = (borderBuffer - 1), NAonly = FALSE) #apply focal matrix to each band
    focal1 = focal(rband, w = psf_Gaussian_Unity, na.rm = TRUE,NAonly = FALSE) #apply focal matrix to each band

    #attribute the point object created earlier with the values of the focal results for each band in the loop
    aggPoints_gaus <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
    rasValue = extract(focal1, aggPoints_gaus, method = 'simple')
    combinePointValue = cbind(aggPoints_gaus,rasValue)
    remove (rasValue)

    # Convert attributed points back to a raster
    rGaussian <- rasterFromXYZ(combinePointValue)
    tempBrick1 <- addLayer(tempBrick1, rGaussian)
    rm (rGaussian)
    #garbage collection
    gc()
  }

  tempBrick1 <- dropLayer(tempBrick1,1)
  outputRasterBrick <- tempBrick1
  proj4string(outputRasterBrick) <- CRS(coordinatesInput)
}

##########################################################################################################################
# If the user requests aggregated resampling then do the following

if (resamplingMethod == "Aggregated") {
  #create a copy of the input raster brick for resampling using the Aggregated method
  rBrickInputOriginalAgg <- rBrickInput
  # produce an aggregated image
  #convert the aggreagate multi band  (raster brick) to a GeoTiff file
  tempBrick1 <- subset (rBrickInputOriginalAgg,1)#temporarily create a raster that matches the extent and projection info of the input data
  tempBrick1 <- aggregate (tempBrick1, fact = (CellSizeDesRatio))
  for (i in 1:nlayers(rBrickInputOriginalAgg)) {
    rband <- subset (rBrickInputOriginalAgg, i)
    rband <- aggregate (rband, fact = (CellSizeDesRatio))
    tempBrick1 <- addLayer(tempBrick1, rband)
  }
  rasterBrickAgg <- dropLayer(tempBrick1, 1)
  proj4string(rasterBrickAgg) <-
```

```
    CRS(coordinatesInput)

  #rename the raster brick for spectral resampling
  outputRasterBrick <- rasterBrickAgg
}

#########################################################################################################
# If the user requests truncated Sinc function resampling then do the following
#Defining the PSF by a sinc Function...

if (resamplingMethod == "Sinc") {
  if (CellSizeDesRatio == 10) {
    SincScalingFactor = 9.93 # set the scaling factor for the Sinc function to ensure that the FWHM is
    # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
    # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
    # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
    y <- x <- seq(-15, 15, length = 31) # length is changing the sampling interval
    y <- x <-  seq(-9, 9, length = 19) # length is changing the sampling interval
    f <- function(x,y) {
      r <- sqrt(x ^ 2 + y ^ 2); 1 * (sin(pi / SincScalingFactor * r)) / ((pi / SincScalingFactor) *
                                                                           r)
      # the following changes the NaN value(s) calculated in the sinc function that should be (as the nature of sinc funtion)
      #be equal to 1
      psf_sinc <- outer(x, y, f)

      psf_sinc <- as.data.frame(psf_sinc)
      dim <- dim(psf_sinc)
      y1 <- unlist(psf_sinc)
      y1[is.nan(y1)] <- 1
      psf_sinc <- matrix(y1 , dim)
      # truncate the sinc function at 0...
      psf_sinc <- pmax(psf_sinc, 0)

      #Alter values of matrix so that they sum to 1 thus maintaining the range of values in the high res imagery psf_sinc = psf_sinc/sum(psf_sinc)
      # for display purposes expand (pad) the matrix with 11 vlaues on the sides so that the gaussian and sinc 3D displays are the same scale
      # pad values on left and right sides
      require(ptw)
      psf_sinc_pad = padzeros(psf_sinc, 6, side = "both")
      # pad vlaues on top and bottom
      psf_sinc_row <- matrix(0, ncol = 31)
      psf_sinc_exp <- rbind(0, psf_sinc_pad, 0)
      psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
      psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
      psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
      psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
      psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
      y <- x <- seq(-16, 16, length = 31)
      persp3D(
        x, y, psf_sinc_exp, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Sinc", curtain = FALSE, border = "black", colkey = FALSE, zlab = "PSF", ticktype =
"detailed", zlim = c(-0.2, 1)
      )
    }

    if (CellSizeDesRatio == 15) {
      SincScalingFactor = 12.4 # set the scaling factor for the Sinc function to ensure that the FWHM is
      # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
      # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
      # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
      y <- x <-  seq(-20, 20, length = 41) # length is changing the sampling interval
      y <- x <- seq(-15, 15, length = 31) # THese values are less than needed. THis is because the Sinc fuction needs to be truncated to prevent the second phase
appearing in the PSF.

      f <- function(x,y) {
        r <- sqrt(x ^ 2 + y ^ 2); 1 * (sin(pi / SincScalingFactor * r)) / ((pi / SincScalingFactor) *
                                                                             r)
      }# 1 is the vertical scale

      # the following changes the NaN value(s) calculated in the sinc function that should be (as the nature of sinc funtion)
      #be equal to 1
      psf_sinc <- outer(x, y, f)

      psf_sinc <- as.data.frame(psf_sinc)
      dim <- dim(psf_sinc)
      y1 <- unlist(psf_sinc)
      y1[is.nan(y1)] <- 1
      psf_sinc <- matrix(y1 , dim)
      # truncate the sinc function at 0...
      psf_sinc <- pmax(psf_sinc, 0)

      #Alter values of matrix so that they sum to 1 thus maintaining the range of values in the high res imagery psf_sinc = psf_sinc/sum(psf_sinc)
```

```
    # for display purposes expand (pad) the matrix with 11 vlaues on the sides so that the gaussian and sinc 3D displays are the same scale
    # pad values on left and right sides
    require(ptw)
    psf_sinc_pad = padzeros(psf_sinc, 5, side = "both")
    # pad vlaues on top and bottom
    psf_sinc_row <- matrix(0, ncol = 31)
    psf_sinc_exp <- rbind(0, psf_sinc_pad, 0)
    psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
    psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
    psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
    psf_sinc_exp <- rbind(0, psf_sinc_exp, 0)
    y <- x <- seq(-20, 20, length = 41)
    persp3D(x, y, psf_sinc_exp, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Sinc", curtain = FALSE, border = "black", colkey = FALSE, zlab = "PSF",
ticktype = "detailed", zlim = c(-0.2, 1))
  }
  # make all values in the PSF sum to equal one (thus retaining the original reflectance values of the area the PSF covers)
  psf_sinc_Unity = psf_sinc_exp / sum(psf_sinc_exp)

  #Apply Sinc kernel matrix to imagery
  #create a loop selecting each of the bands and applying a focal matrix (kernel)
  tempBrick1 <- subset(rBrickInput, 1) # temporarily subset the first band to create a raster brick to as a starting point
  tempBrick1 <- focal(tempBrick1, w = psf_sinc_Unity, na.rm = FALSE,  pad = TRUE, padvalue = ((borderBuffer) - 1), NAonly = FALSE) #apply focal matrix to each band
  aggPoints_sinc <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
  rasValue = extract(tempBrick1, aggPoints_sinc, method = 'simple')
  combinePointValue = cbind(aggPoints_sinc,rasValue)
  remove (rasValue)
  tempBrick1 <- rasterFromXYZ(combinePointValue)
  # loop through each of the bands starting from the second band and writing to the first band sucessively
  for (i in 1:nlayers(rBrickInput)) {
    rband <- subset (rBrickInput, i)
    focal1 = focal(rband, w = psf_sinc_Unity, na.rm = FALSE, pad = TRUE, padvalue = (borderBuffer - 1), NAonly = FALSE) #apply focal matrix to each band
    #attribute the point object created earlier with the values of the focal results for each band in the loop
    aggPoints_sinc <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
    rasValue = extract(focal1, aggPoints_sinc, method = 'simple')
    combinePointValue = cbind(aggPoints_sinc,rasValue)
    remove (rasValue)

    # Convert attributed points back to a raster
    rSinc <- rasterFromXYZ(combinePointValue)
    tempBrick1 <- addLayer(tempBrick1, rSinc)
  }

  tempBrick1 <- dropLayer(tempBrick1,1)
  outputRasterBrick <- tempBrick1
  proj4string(outputRasterBrick) <- CRS(coordinatesInput)
 }
}
# remove the rasterBrickNoBlur as it is no longer needed
rm (rasterBrickNoBlur)
#garbage collection
gc()
#########################################################################################################################
## If the user requests sinc Hamming resampling then do the following
if (resamplingMethod == "Sinc Hamming") {
 #Defining the PSF by a sinc Function modified to hamming function..
 #First define the Sinc PSF...
 if (CellSizeDesRatio == 10) {
   SincScalingFactor = 9.93 # set the scaling factor for the Sinc function to ensure that the FWHM is
   # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
   # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
   # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
   y <- x <- seq(-15, 15, length = 31)
 }
 if (CellSizeDesRatio == 15) {
   SincScalingFactor = 16.0 # set the scaling factor for the Sinc function to ensure that the FWHM is
   # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
   # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
   # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
   y <- x <- seq(-20, 20, length = 41)
 }

 f <- function(x,y) {
   r <-
     sqrt(x ^ 2 + y ^ 2); 1 * (sin(pi / SincScalingFactor * r)) / ((pi / SincScalingFactor) * r)  # 1 is the vertical scale
   #r[x == 0] <- 1
 }

 psf_sinc_4Hamming <- outer(x, y, f)

 # change the central NaN value to 1...
```

```
psf_sinc_4Hamming[is.nan(psf_sinc_4Hamming)] <- 1

# Now define the Hamming LSF function...
if (CellSizeDesRatio == 10) {
  Hamming <-x <- seq(1, 32, length = 31) # length is changing the sampling interval
  f <- function(x) {
    y <- Hamming <- 0.54 - 0.46 * cos (2 * pi * (x) / 33)  # 30 is double the Sinc truncated for
  }
}
if (CellSizeDesRatio == 15) {
  Hamming <-x <- seq(1, 42, length = 41) # length is changing the sampling interval
  f <- function(x) {
    y <- Hamming <- 0.54 - 0.46 * cos (2 * pi * (x) / 43)  # 30 is double the Sinc truncated for
  }
}
Hamming1 = f (x)

if (CellSizeDesRatio == 10) {
  #Convert the LSF (1D) to a PSF (2D) by calculating the values row by row...
  HammingR1 = Hamming1 / max(Hamming1) * Hamming1[1]
  HammingR2 = Hamming1 / max(Hamming1) * Hamming1[2]
  HammingR3 = Hamming1 / max(Hamming1) * Hamming1[3]
  HammingR4 = Hamming1 / max(Hamming1) * Hamming1[4]
  HammingR5 = Hamming1 / max(Hamming1) * Hamming1[5]
  HammingR6 = Hamming1 / max(Hamming1) * Hamming1[6]
  HammingR7 = Hamming1 / max(Hamming1) * Hamming1[7]
  HammingR8 = Hamming1 / max(Hamming1) * Hamming1[8]
  HammingR9 = Hamming1 / max(Hamming1) * Hamming1[9]
  HammingR10 = Hamming1 / max(Hamming1) * Hamming1[10]
  HammingR11 = Hamming1 / max(Hamming1) * Hamming1[11]
  HammingR12 = Hamming1 / max(Hamming1) * Hamming1[12]
  HammingR13 = Hamming1 / max(Hamming1) * Hamming1[13]
  HammingR14 = Hamming1 / max(Hamming1) * Hamming1[14]
  HammingR15 = Hamming1 / max(Hamming1) * Hamming1[15]
  HammingR16 = Hamming1 / max(Hamming1) * Hamming1[16]
  HammingR17 = Hamming1 / max(Hamming1) * Hamming1[17]
  HammingR18 = Hamming1 / max(Hamming1) * Hamming1[18]
  HammingR19 = Hamming1 / max(Hamming1) * Hamming1[19]
  HammingR20 = Hamming1 / max(Hamming1) * Hamming1[20]
  HammingR21 = Hamming1 / max(Hamming1) * Hamming1[21]
  HammingR22 = Hamming1 / max(Hamming1) * Hamming1[22]
  HammingR23 = Hamming1 / max(Hamming1) * Hamming1[23]
  HammingR24 = Hamming1 / max(Hamming1) * Hamming1[24]
  HammingR25 = Hamming1 / max(Hamming1) * Hamming1[25]
  HammingR26 = Hamming1 / max(Hamming1) * Hamming1[26]
  HammingR27 = Hamming1 / max(Hamming1) * Hamming1[27]
  HammingR28 = Hamming1 / max(Hamming1) * Hamming1[28]
  HammingR29 = Hamming1 / max(Hamming1) * Hamming1[29]
  HammingR30 = Hamming1 / max(Hamming1) * Hamming1[30]
  HammingR31 = Hamming1 / max(Hamming1) * Hamming1[31]

  # assemble the rows to complete a matrix representing the PSF
  psf_Hamming = matrix(c(HammingR1, HammingR2, HammingR3, HammingR4, HammingR5, HammingR6, HammingR7, HammingR8, HammingR9,HammingR11,
HammingR12, HammingR13, HammingR14, HammingR15, HammingR16, HammingR17, HammingR18, HammingR19,HammingR20, HammingR21, HammingR22,
HammingR23, HammingR24, HammingR25, HammingR26, HammingR27, HammingR28, HammingR29,HammingR30, HammingR31), nrow = 31, ncol = 31)
}

if (CellSizeDesRatio == 15) {
  #Convert the LSF (1D) to a PSF (2D) by calculating the values row by row...
  HammingR1 = Hamming1 / max(Hamming1) * Hamming1[1]
  HammingR2 = Hamming1 / max(Hamming1) * Hamming1[2]
  HammingR3 = Hamming1 / max(Hamming1) * Hamming1[3]
  HammingR4 = Hamming1 / max(Hamming1) * Hamming1[4]
  HammingR5 = Hamming1 / max(Hamming1) * Hamming1[5]
  HammingR6 = Hamming1 / max(Hamming1) * Hamming1[6]
  HammingR7 = Hamming1 / max(Hamming1) * Hamming1[7]
  HammingR8 = Hamming1 / max(Hamming1) * Hamming1[8]
  HammingR9 = Hamming1 / max(Hamming1) * Hamming1[9]
  HammingR10 = Hamming1 / max(Hamming1) * Hamming1[10]
  HammingR11 = Hamming1 / max(Hamming1) * Hamming1[11]
  HammingR12 = Hamming1 / max(Hamming1) * Hamming1[12]
  HammingR13 = Hamming1 / max(Hamming1) * Hamming1[13]
  HammingR14 = Hamming1 / max(Hamming1) * Hamming1[14]
  HammingR15 = Hamming1 / max(Hamming1) * Hamming1[15]
  HammingR16 = Hamming1 / max(Hamming1) * Hamming1[16]
  HammingR17 = Hamming1 / max(Hamming1) * Hamming1[17]
  HammingR18 = Hamming1 / max(Hamming1) * Hamming1[18]
  HammingR19 = Hamming1 / max(Hamming1) * Hamming1[19]
  HammingR20 = Hamming1 / max(Hamming1) * Hamming1[20]
  HammingR21 = Hamming1 / max(Hamming1) * Hamming1[21]
```

```
    HammingR22 = Hamming1 / max(Hamming1) * Hamming1[22]
    HammingR23 = Hamming1 / max(Hamming1) * Hamming1[23]
    HammingR24 = Hamming1 / max(Hamming1) * Hamming1[24]
    HammingR25 = Hamming1 / max(Hamming1) * Hamming1[25]
    HammingR26 = Hamming1 / max(Hamming1) * Hamming1[26]
    HammingR27 = Hamming1 / max(Hamming1) * Hamming1[27]
    HammingR28 = Hamming1 / max(Hamming1) * Hamming1[28]
    HammingR29 = Hamming1 / max(Hamming1) * Hamming1[29]
    HammingR30 = Hamming1 / max(Hamming1) * Hamming1[30]
    HammingR31 = Hamming1 / max(Hamming1) * Hamming1[31]
    HammingR32 = Hamming1 / max(Hamming1) * Hamming1[32]
    HammingR33 = Hamming1 / max(Hamming1) * Hamming1[33]
    HammingR34 = Hamming1 / max(Hamming1) * Hamming1[34]
    HammingR35 = Hamming1 / max(Hamming1) * Hamming1[35]
    HammingR36 = Hamming1 / max(Hamming1) * Hamming1[36]
    HammingR37 = Hamming1 / max(Hamming1) * Hamming1[37]
    HammingR38 = Hamming1 / max(Hamming1) * Hamming1[38]
    HammingR39 = Hamming1 / max(Hamming1) * Hamming1[39]
    HammingR40 = Hamming1 / max(Hamming1) * Hamming1[40]
    HammingR41 = Hamming1 / max(Hamming1) * Hamming1[41]

    # assemble the rows to complete a matrix representing the PSF
    psf_Hamming = matrix(c(HammingR1, HammingR2, HammingR3, HammingR4, HammingR5, HammingR6, HammingR7, HammingR8, HammingR9,HammingR11,
HammingR12, HammingR13, HammingR14, HammingR15, HammingR16, HammingR17, HammingR18, HammingR19,HammingR20, HammingR21, HammingR22,
HammingR23, HammingR24, HammingR25, HammingR26, HammingR27, HammingR28, HammingR29,HammingR30, HammingR31, HammingR32, HammingR33,
HammingR34, HammingR35, HammingR36, HammingR37, HammingR38, HammingR39,HammingR40, HammingR41), nrow = 41, ncol = 41)
  }

  #Combine the hamming PSF with the Sinc PSF to get a PSF that truncates at zero smoothly.
  psf_SincHam <- psf_Hamming * psf_sinc_4Hamming
#   #plot the PSFs in 3D
#   persp3D(x, y, psf_Hamming, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Sinc - Hamming", curtain = FALSE, border = "black", colkey = FALSE, zlab =
"PSF", ticktype = "detailed", zlim = c(-0.2, 1))
#   persp3D(x, y, psf_SincHam, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Sinc - Hamming", curtain = FALSE, border = "black", colkey = FALSE, zlab =
"PSF", ticktype = "detailed", zlim = c(-0.2, 1))
#
  # make all values in the PSF sum to equal one (thus retaining the original reflectance values of the area the PSF covers)
  psf_SincHam_Unity = psf_SincHam / sum(psf_SincHam)

  #Apply Sinc - Hamming kernel matrix to imagery
  #create a loop selecting each of the bands and applying a focal matrix (kernel)
  tempBrick1 <- subset(rBrickInput, 1) # temporarily subset the first band to create a raster brick to as a starting point
  tempBrick1 <- focal(tempBrick1, w = psf_SincHam_Unity, na.rm = FALSE,  pad = TRUE, padvalue = (borderBuffer - 1), NAonly = FALSE) #apply focal matrix to each
band
  aggPoints_sinc <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
  rasValue = extract(tempBrick1, aggPoints_sinc, method = 'simple')
  combinePointValue = cbind(aggPoints_sinc,rasValue)
  remove (rasValue)
  tempBrick1 <- rasterFromXYZ(combinePointValue)
  # loop through each of the bands starting from the second band and writing to the first band sucessively
  for (i in 1:nlayers(rBrickInput)) {
    rband <- subset (rBrickInput, i)
    focal1 = focal(rband, w = psf_SincHam_Unity, na.rm = FALSE,  pad = TRUE, padvalue = (borderBuffer - 1), NAonly = FALSE) #apply focal matrix to each band
    #attribute the point object created earlier with the values of the focal results for each band in the loop
    aggPoints_sinc <- aggPoints[,-(3),drop = FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
    rasValue = extract(focal1, aggPoints_sinc, method = 'simple')
    combinePointValue = cbind(aggPoints_sinc,rasValue)
    remove (rasValue)
    # Convert attributed points back to a raster
    rSincHam <- rasterFromXYZ(combinePointValue)
    tempBrick1 <- addLayer(tempBrick1, rSincHam)
  }

  tempBrick1 <- dropLayer(tempBrick1,1)
  outputRasterBrick <- tempBrick1
  proj4string(outputRasterBrick) <-
    CRS(coordinatesInput)
}

#########################################################################################################################
## If the user requests sinc Blackman resampling then do the following...
#resamplingMethod = "Sinc Blackman" # insert a # if a Gaussian window kernel is not desired *
if (resamplingMethod == "Sinc Blackman") {
  # first create a LSF of the Blackman function...
  #Defining the PSF by a sinc Function modified to Blackman function..
  #First define the Sinc PSF...
  if (CellSizeDesRatio == 10) {
    SincScalingFactor = 12.35 # set the scaling factor for the Sinc function to ensure that the FWHM is
    # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
    # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
```

```
  # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
  y <- x <- seq(-15, 15, length=31)
 }
 if (CellSizeDesRatio == 15) {
  SincScalingFactor = 24.3 # set the scaling factor for the Sinc function to ensure that the FWHM is
  # sampled correctly. This value is the factor for a FWHM measured from 0 to 1. Switch this value to 8.67946 for the
  # FWHM measured from the minimum value (negetaive value) to max. These values were derived from sampling the
  # Sinc function curve with a vertical axis scaled from 0-1 scaled from 0-1 in the
  y <- x <- seq(-20, 20, length=41)
 }

 f <- function(x,y) {
  r <- sqrt(x^2+y^2); 1 * (sin(pi/SincScalingFactor*r)) / ((pi/SincScalingFactor)*r)  # 1 is the vertical scale
  #r[x == 0] <- 1
 }
 psf_sinc_4Blackman <- outer(x, y, f)

 #define the Blackman PSF
 if (CellSizeDesRatio == 10) {
  Blackman <- x <- seq(1, 32, length=31) # length is changing the sampling interval
  f <- function(x) {
    y <- Blackman <- 0.42 - 0.5 * cos (2*pi*(x) / 33) + 0.08 * cos(4*pi*(x)/33)
  }
 }

 if (CellSizeDesRatio == 15) {
  Blackman <- x <- seq(1, 42, length=41) # length is changing the sampling interval
  f <- function(x) {
    y <- Blackman <- 0.42 - 0.5 * cos (2*pi*(x) / 43) + 0.08 * cos(4*pi*(x)/43)
  }
 }
 Blackman1 = f (x)

 if (CellSizeDesRatio == 10) {
  #Convert the LSF (1D) to a PSF (2D) by calculating the values row by row...
  BlackmanR1 = Blackman1/max(Blackman1) * Blackman1[1]
  BlackmanR2 = Blackman1/max(Blackman1) * Blackman1[2]
  BlackmanR3 = Blackman1/max(Blackman1) * Blackman1[3]
  BlackmanR4 = Blackman1/max(Blackman1) * Blackman1[4]
  BlackmanR5 = Blackman1/max(Blackman1) * Blackman1[5]
  BlackmanR6 = Blackman1/max(Blackman1) * Blackman1[6]
  BlackmanR7 = Blackman1/max(Blackman1) * Blackman1[7]
  BlackmanR8 = Blackman1/max(Blackman1) * Blackman1[8]
  BlackmanR9 = Blackman1/max(Blackman1) * Blackman1[9]
  BlackmanR10 = Blackman1/max(Blackman1) * Blackman1[10]
  BlackmanR11 = Blackman1/max(Blackman1) * Blackman1[11]
  BlackmanR12 = Blackman1/max(Blackman1) * Blackman1[12]
  BlackmanR13 = Blackman1/max(Blackman1) * Blackman1[13]
  BlackmanR14 = Blackman1/max(Blackman1) * Blackman1[14]
  BlackmanR15 = Blackman1/max(Blackman1) * Blackman1[15]
  BlackmanR16 = Blackman1/max(Blackman1) * Blackman1[16]
  BlackmanR17 = Blackman1/max(Blackman1) * Blackman1[17]
  BlackmanR18 = Blackman1/max(Blackman1) * Blackman1[18]
  BlackmanR19 = Blackman1/max(Blackman1) * Blackman1[19]
  BlackmanR20 = Blackman1/max(Blackman1) * Blackman1[20]
  BlackmanR21 = Blackman1/max(Blackman1) * Blackman1[21]
  BlackmanR22 = Blackman1/max(Blackman1) * Blackman1[22]
  BlackmanR23 = Blackman1/max(Blackman1) * Blackman1[23]
  BlackmanR24 = Blackman1/max(Blackman1) * Blackman1[24]
  BlackmanR25 = Blackman1/max(Blackman1) * Blackman1[25]
  BlackmanR26 = Blackman1/max(Blackman1) * Blackman1[26]
  BlackmanR27 = Blackman1/max(Blackman1) * Blackman1[27]
  BlackmanR28 = Blackman1/max(Blackman1) * Blackman1[28]
  BlackmanR29 = Blackman1/max(Blackman1) * Blackman1[29]
  BlackmanR30 = Blackman1/max(Blackman1) * Blackman1[30]
  BlackmanR31 = Blackman1/max(Blackman1) * Blackman1[31]

  # assemble the rows to complete a matrix representing the PSF
  psf_Blackman = matrix(c(BlackmanR1, BlackmanR2, BlackmanR3, BlackmanR4, BlackmanR5, BlackmanR6, BlackmanR7, BlackmanR8, BlackmanR9, BlackmanR10,
BlackmanR11, BlackmanR12, BlackmanR13, BlackmanR14, BlackmanR15, BlackmanR16, BlackmanR17, BlackmanR18, BlackmanR19,BlackmanR20, BlackmanR21,
BlackmanR22, BlackmanR23, BlackmanR24, BlackmanR25, BlackmanR26, BlackmanR27, BlackmanR28, BlackmanR29,BlackmanR30, BlackmanR31), nrow = 31, ncol
= 31)
  y <- x <- seq(-15, 15, length = 31)
 }

 if (CellSizeDesRatio == 15) {
  #Convert the LSF (1D) to a PSF (2D) by calculating the values row by row...
  BlackmanR1 = Blackman1/max(Blackman1) * Blackman1[1]
  BlackmanR2 = Blackman1/max(Blackman1) * Blackman1[2]
  BlackmanR3 = Blackman1/max(Blackman1) * Blackman1[3]
```

```
  BlackmanR4 = Blackman1/max(Blackman1) * Blackman1[4]
  BlackmanR5 = Blackman1/max(Blackman1) * Blackman1[5]
  BlackmanR6 = Blackman1/max(Blackman1) * Blackman1[6]
  BlackmanR7 = Blackman1/max(Blackman1) * Blackman1[7]
  BlackmanR8 = Blackman1/max(Blackman1) * Blackman1[8]
  BlackmanR9 = Blackman1/max(Blackman1) * Blackman1[9]
  BlackmanR10 = Blackman1/max(Blackman1) * Blackman1[10]
  BlackmanR11 = Blackman1/max(Blackman1) * Blackman1[11]
  BlackmanR12 = Blackman1/max(Blackman1) * Blackman1[12]
  BlackmanR13 = Blackman1/max(Blackman1) * Blackman1[13]
  BlackmanR14 = Blackman1/max(Blackman1) * Blackman1[14]
  BlackmanR15 = Blackman1/max(Blackman1) * Blackman1[15]
  BlackmanR16 = Blackman1/max(Blackman1) * Blackman1[16]
  BlackmanR17 = Blackman1/max(Blackman1) * Blackman1[17]
  BlackmanR18 = Blackman1/max(Blackman1) * Blackman1[18]
  BlackmanR19 = Blackman1/max(Blackman1) * Blackman1[19]
  BlackmanR20 = Blackman1/max(Blackman1) * Blackman1[20]
  BlackmanR21 = Blackman1/max(Blackman1) * Blackman1[21]
  BlackmanR22 = Blackman1/max(Blackman1) * Blackman1[22]
  BlackmanR23 = Blackman1/max(Blackman1) * Blackman1[23]
  BlackmanR24 = Blackman1/max(Blackman1) * Blackman1[24]
  BlackmanR25 = Blackman1/max(Blackman1) * Blackman1[25]
  BlackmanR26 = Blackman1/max(Blackman1) * Blackman1[26]
  BlackmanR27 = Blackman1/max(Blackman1) * Blackman1[27]
  BlackmanR28 = Blackman1/max(Blackman1) * Blackman1[28]
  BlackmanR29 = Blackman1/max(Blackman1) * Blackman1[29]
  BlackmanR30 = Blackman1/max(Blackman1) * Blackman1[30]
  BlackmanR31 = Blackman1/max(Blackman1) * Blackman1[31]
  BlackmanR32 = Blackman1/max(Blackman1) * Blackman1[32]
  BlackmanR33 = Blackman1/max(Blackman1) * Blackman1[33]
  BlackmanR34 = Blackman1/max(Blackman1) * Blackman1[34]
  BlackmanR35 = Blackman1/max(Blackman1) * Blackman1[35]
  BlackmanR36 = Blackman1/max(Blackman1) * Blackman1[36]
  BlackmanR37 = Blackman1/max(Blackman1) * Blackman1[37]
  BlackmanR38 = Blackman1/max(Blackman1) * Blackman1[38]
  BlackmanR39 = Blackman1/max(Blackman1) * Blackman1[39]
  BlackmanR40 = Blackman1/max(Blackman1) * Blackman1[40]
  BlackmanR41 = Blackman1/max(Blackman1) * Blackman1[41]

  # assemble the rows to complete a matrix representing the PSF
  psf_Blackman = matrix(c(BlackmanR1, BlackmanR2, BlackmanR3, BlackmanR4, BlackmanR5, BlackmanR6, BlackmanR7, BlackmanR8, BlackmanR9, BlackmanR10,
BlackmanR11, BlackmanR12, BlackmanR13, BlackmanR14, BlackmanR15, BlackmanR16, BlackmanR17, BlackmanR18, BlackmanR19,BlackmanR20, BlackmanR21,
BlackmanR22, BlackmanR23, BlackmanR24, BlackmanR25, BlackmanR26, BlackmanR27, BlackmanR28, BlackmanR29,BlackmanR30, BlackmanR31, BlackmanR32,
BlackmanR33, BlackmanR34, BlackmanR35, BlackmanR36, BlackmanR37, BlackmanR38, BlackmanR39,BlackmanR40, BlackmanR41), nrow = 41, ncol = 41)
  y <- x <- seq(-20, 20, length = 41)
 }

 #Combine the Blackman PSF with the Sinc PSF to get a PSF that truncates at zero smoothly...
 psf_SincBla <- psf_Blackman * psf_sinc_4Blackman
 psf_SincBla[is.na(psf_SincBla)] <- 1
 # make all values in the PSF sum to equal ove (thus retaining the original reflectance values of the area the PSF covers)
 psf_SincBla_Unity <- (psf_SincBla) / sum(psf_SincBla)

#  #plot the PSFs in 3D
#  persp3D(x, y, psf_Blackman, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Blackman", curtain = FALSE, border = "black", colkey = FALSE, zlab =
"PSF", ticktype = "detailed", zlim = c(-0.2, 1))
#  persp3D(x, y, psf_SincBla, theta = 20, phi = 15 , expand = .6, facets = TRUE,    main = "Sinc - Blackman", curtain = FALSE, border = "black", colkey = FALSE, zlab =
"PSF", ticktype = "detailed", zlim = c(-0.2, 1))
#
 ##################################################################################################################
 #Apply Sinc - Blackman kernel matrix to imagery
 #create a loop selecting each of the bands and applying a focal matrix (kernel)
 tempBrick1 <- subset(rBrickInput, 1) # temporarily subset the first band to create a raster brick to as a starting point
 tempBrick1 <- focal(tempBrick1, w=psf_SincBla_Unity, na.rm=FALSE,  pad=TRUE, padvalue=(borderBuffer-1), NAonly=FALSE) #apply focal matrix to each band
 aggPoints_sinc <- aggPoints[,-(3),drop=FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
 rasValue=extract(tempBrick1, aggPoints_sinc, method='simple')
 combinePointValue=cbind(aggPoints_sinc,rasValue)
 remove (rasValue)
 rm (tempBrick1)
 #garbage collection
 gc()
 outputRasterBrick <- rasterFromXYZ(combinePointValue)
 # loop through each of the bands starting from the second band and writing to the first band sucessively
 for (i in 1:nlayers(rBrickInput)){
  rband <- subset (rBrickInput, i)
  focal1 = focal(rband, w=psf_SincBla_Unity, na.rm=FALSE, pad=TRUE, padvalue=(borderBuffer-1), NAonly=FALSE) #apply focal matrix to each band
  #attribute the point object created earlier with the values of the focal results for each band in the loop
  aggPoints_sinc <- aggPoints[,-(3),drop=FALSE] #drop the third column as its not needed and will cause an errror when attempting to extract raster values
  rasValue=extract(focal1, aggPoints_sinc, method='simple')
  combinePointValue=cbind(aggPoints_sinc,rasValue)
```

```
    remove (rasValue)
    remove (focal1)

    # Convert attributed points back to a raster
    rSincBla <- rasterFromXYZ(combinePointValue)
    outputRasterBrick <- addLayer(outputRasterBrick, rSincBla)
  }

  outputRasterBrick <- dropLayer(outputRasterBrick,1)
  proj4string(outputRasterBrick) <- CRS(coordinatesInput)
}

# Move the raster data back to its corrected X and Y postion. It was previously moved upwards and to the right to account for the mirroring of values
#outputRasterBrick <- raster::shift(outputRasterBrick, x= -(borderBuffer), y= -(borderBuffer))
#crop data back to the original dimensions (it was shifted up and the right to create mirrors edges)
# xminextent = xmin(rBrickInputOriginal) - (borderBuffer)
# xmaxextent = xmax(rBrickInputOriginal) + (borderBuffer)
# yminextent = ymin(rBrickInputOriginal) - (borderBuffer)
# ymaxextent = ymax(rBrickInputOriginal) + (borderBuffer)
xminextent = xmin(rBrickInputOriginal)
xmaxextent = xmax(rBrickInputOriginal)
yminextent = ymin(rBrickInputOriginal)
ymaxextent = ymax(rBrickInputOriginal)
e <- extent(xminextent, xmaxextent, yminextent, ymaxextent)
#outputRasterBrick <- crop(outputRasterBrick, e)
#outputRasterBrick <- raster::shift(outputRasterBrick, x = -(borderBuffer), y = -(borderBuffer))
rm (rBrickInputOriginal)
#garbage collection
gc()

#convert the convolved spatially raster brick to a GeoTiff file
outputfilename <- "Output_Spatially_Convolved.tif"
outputFile <- writeRaster(outputRasterBrick, filename = (outputfilename), overwrite = TRUE)

#remove the rBrickInput as it not needed any more
rm (rBrickInput)
#garbage collection
gc()

#remove all objects that are matched by 'rband' as they are not longer needed
rm (list=ls(pattern=glob2rx("rband*")))
#remove the OutputRaster as it not needed after outputing the file to a TIF image
rm (outputFile)
#rm (tempBrick1)
#garbage collection
gc()

#########################################################################################################################################
# Create an aggregated PSF for display purposes
psf_agg <- matrix(1,nrow = 15, ncol = 15)
# pad values on left and right sides
require(ptw)
psf_agg_pad = padzeros(psf_agg, 8, side="both")

# pad vlaues on top and bottom
psf_agg_exp <- rbind(0, psf_agg_pad, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_exp <- rbind(0, psf_agg_exp, 0)
psf_agg_norm <- psf_agg_exp
y <- x <-  seq(-20, 20, length = 41)

#proc.time() - starttime
#print (proc.time() - starttime)
# end of script
```

```
#################################################################################################
# 3. Spectral Resample.R
#
# Script details:
# This script spectrally convolves input hyperspectral data regardless of its spectral
# resolution to a lower desired spectral resolution. It does this for two independent sensors
# (VNIR & SWIR) measuring at different spectral resolutions. To achieve this, the script
# repeats three key steps twice, once for each sensor: 1) it reads the input data's
# wavelengths, 2) it produces a 1 nm spectral super-resolution by linearly interpolating the
# input data, and 3) it convolves this super resolution using Gaussian weighting to the
# desired spectral resolution.
#
# Users are directed to read the output text (printed to on the screen) at the end of running
# the script as there are manual steps that have to be undertaken to complete the process.
# This specifically is the wavelength information that has to be copied to the header files
# for both the VNIR and SWIR sensors.
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada -
# Natural Resources Canada
#
# Script completed December, 2016
#################################################################################################
# load imagery
rBrickSpectral1 <- outputRasterBrick
#rBrickSpectral1 <- rBrickInput2 #temporary line. Delete later. For debugging a subset data
#rBrickSpectral1 <- brick("U:/Thesis/Data/TestData/Output_Gaussian.tif", package="raster")
newRasterBrick <- subset (rBrickSpectral1, 1)

#start a stopwatch to time the script
#starttime <- proc.time()

# convert the wavelengths to nm if stored as um
if (spectralResolution == "um"){
  wavelength <- wavelength * 1000}

# Add a band to be removed later. This is created to overcome the error caused by the last layer
SWIRmaximumWavelength <- SWIRmaximumWavelength + desiredSpectralResSWIR
#################################################################################################
# First the script will create a superresolution image at 1 nm linearly interpolated from the
# input data to spectrally convolve to the desired output resolution. This super spectral
# resolution will be using rounded 1 nm intervals. Sometimes, however, the input data does not
# start with rounded whole values for the spectral resolution. Here the following code creates
# a vector object of these 1 nm values (wavelength band positions) regardless if the input
# data is at fractions of a whole number.
desiredWavelengthWhole1 <- min(wavelength)
# determine the minimum wavelength in the wavelength vector object
desiredWavelengthWhole2 <- max(wavelength)
# determine the maximum wavelength in the wavelength vector object
desiredWavelengthWhole3 <- ceiling((desiredWavelengthWhole1/1) * 1)
# round the minimum value to 1 m
desiredWavelengthWhole4 <- floor(desiredWavelengthWhole2/1) * 1 #round the maximum value to 1 m
desiredWavelengthWhole5 <- seq(desiredWavelengthWhole3, desiredWavelengthWhole4, by=1)
# now you have a vector of 1 nm intervals for the creation of a superresolution image

# create a new R raster brick object that will house the output bands by using the first band
# of the original input data.
newRasterBrickVNIR <- subset (rBrickSpectral1,1)
#################################################################################################
#################################################################################################
#vNIR sensor
#################################################################################################
#################################################################################################
# Find the closest wavelength in the new 1 nm vector list to just below the desired minimum
# spectral range value (420nm for EnMap) while considering the buffer required for gaussian
# weighting of the input bands. This value is 10nm for a spectral resolution of 6.5 nm. In
# other words there is an additional value of spectral resolution required in order to
# convolve the bands to the desired resolution.The amount of buffer is controlled by the shape
# of the guassian sampling distance.
if (desiredSpectralResVNIR == 6.5){
  VNIRBuffer <- 10}
if (desiredSpectralResVNIR == 7.5){
  VNIRBuffer <- 12}
if (desiredSpectralResVNIR == 10){
  VNIRBuffer <- 15}

# reset the VNIR minimum wavelength vector list to consider the necessary buffer
VNIRminimumWavelengthBuff <- VNIRminimumWavelength - (VNIRBuffer + 1)
# We now need to create a table to be used later. The table will be used to create vectors
# of the band order and the associated next wavelenth
for (i in (VNIRminimumWavelengthBuff)){
```

```
 dt <- data.table(wavelength, val = wavelength) # create a data table of wavelength
 setattr(dt, "sorted", "wavelength")
 # write a function (called DFshift) that will allow the values in a data table to populate
 # a column with either the following or preceeding values in a table
 DFshift <- function(x, offset = 1, pad = NA) {
  r <- (1 + offset):(length(x) + offset)
  r[r<1] <- NA
  ans <- x[r]
  ans[is.na(ans)] <- pad
  return(ans)
 }
 dt[, D := DFshift(val,-1)]#add a column that contains all the values of the preceeding
 # row. This is necessary for the steps below.
 temp1 <- dt[J(i), roll = "nearest"]
 #wavelengthOrderNumMin <-sapply(i,function(i)which.min(abs(i - wavelength)))
 wavelengthOrderNumMin <-which.min(abs(i - wavelength))
 # change the record number if the closest record is greater than the minimum required EnMap
 # wavelength
 closestWavelength <-subset(temp1, select=c(val))
 if(closestWavelength > i){
  closestWavelength <-subset(temp1, select=c(D)) # use the value from the row above. This
  # is actually stored in the column "D".
 } # change the val (order number) to one record lower than the closest value so that now the
 # subset will be the closest record but lower than the minimum value.
}

# Find the closest wavelength in the new 1 nm vector file to just above the maximum desired
# spectral range value (1000nm for EnMap's VNIR) while considering the buffer required for
# gaussian weighting of the input bands. In other words there is an additional value of
# spectral resolution required in order to conduct convolvion of the bands.
VNIRmaximumWavelengthBuff <- VNIRmaximumWavelength + (VNIRBuffer + 1)
for (i in (VNIRmaximumWavelengthBuff)){
 dt2 <- data.table(wavelength, val = wavelength)
 setattr(dt2, "sorted", "wavelength")
 # add a column that contains all the values of the preceeding row. This is necessary for the
 # steps below.
 dt2[, D := DFshift(val,+1)]
 temp1 <- dt2[J(i), roll = "nearest"]
 wavelengthOrderNumMax <-sapply(i,function(i)which.min(abs(i - wavelength)))
 wavelengthOrderNumMax <-which.min(abs(i - wavelength))
 closestMaxWavelength <-subset(temp1, select=c(val))
 # change the record number if the closest record is greater than the minimum required
 # wavelength
 if(closestMaxWavelength < i){
  #use the value from the row above. This is actually stored in the column "D"
  closestMaxWavelength <-subset(temp1, select=c(D))
 } # change the val (order number) to one record lower than the closest value so that now the
  # subset will be the closest record but lower than the minimum value.
}

# subset the VNIR bands from the input data
nlayersMinus1 <- wavelengthOrderNumMin - 1
#nlayersMinus2 <- wavelengthOrderNumMin - 2
if (nlayersMinus1 < 0){
 nlayersMinus1 <- 1
}
nlayersPlus1 <- wavelengthOrderNumMax + 1
# drop the higher bands than those needed for processing.
rasterBrickSpectralVNIR <- dropLayer(rBrickSpectral1,c(nlayersPlus1:nlayers(rBrickSpectral1)))
# drop the lower bands than those needed for processing.
rasterBrickSpectralVNIR <- dropLayer(rasterBrickSpectralVNIR,c(0:nlayersMinus1))

# Create a vector containing the corresponding wavelengths for the VNIR raster Brick bands of
# the desired output
# Strip the last records from the wavelength vector
wavelengthVNIR <- head(wavelength, (wavelengthOrderNumMax))
wavelengthLength <- length(wavelengthVNIR) - wavelengthOrderNumMin + 1
# Strip the first records from the wavelength vector
wavelengthVNIR <- tail(wavelengthVNIR, (wavelengthLength))

#############################################################################################
# before interpolating strip out the first bands below the minimum spectral range value
# determined at the start of this script. This was user defined earlier.
desiredWavelengthWhole5backup <- desiredWavelengthWhole5
while (min(desiredWavelengthWhole5) < VNIRminimumWavelengthBuff) {
 desiredWavelengthWhole5 <- tail(desiredWavelengthWhole5, length(desiredWavelengthWhole5) - 1)}

while (max(desiredWavelengthWhole5) > VNIRmaximumWavelengthBuff) {
 desiredWavelengthWhole5 <- head(desiredWavelengthWhole5, length(desiredWavelengthWhole5) - 1)}
```

```
#################################################################################################
#linearly interpolate each desired wavelength to create 1 nm super resolution image
for (i in (desiredWavelengthWhole5)){
 # find the nearest wavelength to the desired wavelength and determine which record this is
 # in the vector
 dt3 <- data.table(wavelengthVNIR, val = wavelengthVNIR)
 setattr(dt3, "sorted", "wavelengthVNIR")
 temp1 <- dt3[J(i), roll = "nearest"]
 wavelengthVNIROrderNum <-sapply(i,function(i)which.min(abs(i - wavelengthVNIR)))
 closestWavelength <-subset(temp1, select=c(val))

 # determine the X values for the interpolation equation
 if(closestWavelength < i){
  x1 <-  closestWavelength
  wavelengthVNIROrderNumPlusOne <- (wavelengthVNIROrderNum) + 1
  x2 <-  wavelengthVNIR[(wavelengthVNIROrderNumPlusOne)]

 } else if(closestWavelength == i){
  x1 <-  closestWavelength
  wavelengthVNIROrderNumPlusOne <- (wavelengthVNIROrderNum) + 1
  x2 <-  wavelengthVNIR[(wavelengthVNIROrderNumPlusOne)]

 } else { # when the closest wavelength is greater than the desired wavelength
  wavelengthVNIROrderNumMinusOne <- (wavelengthVNIROrderNum) - 1
  if (wavelengthVNIROrderNum == 1){
   next
  }
  x1 <- wavelengthVNIR[(wavelengthVNIROrderNumMinusOne)]
  x2 <- closestWavelength
 }

 #determine the y values for the interpolation equation
 if(i == min(wavelengthVNIR)){
  y <- subset (rasterBrickSpectralVNIR,1)
 } else if(i == max(wavelengthVNIR)){
  y <- nlayers(rasterBrickSpectralVNIR)
  y <- subset (rasterBrickSpectralVNIR,(y))
 } else if(i == closestWavelength){
  y <- subset (rasterBrickSpectralVNIR,(wavelengthVNIROrderNum))

 } else {   #interpolate the remaining desired wavelengths
  # apply the linear interpolation equation but first set the remaining variables
  x <- i
  if(closestWavelength > i){ # when the closest wavelength is > than the desired wavelength
   wavelengthVNIROrderNumMinusOne <- (wavelengthVNIROrderNum) - 1
   y1 <-  subset(rasterBrickSpectralVNIR, (wavelengthVNIROrderNumMinusOne))
   y2 <-  subset(rasterBrickSpectralVNIR, (wavelengthVNIROrderNum))

  } else #{  # when the closest wavelength is less than the desired wavelength
   wavelengthVNIROrderNumPlusOne <- (wavelengthVNIROrderNum) + 1
   if (wavelengthVNIROrderNumPlusOne < nlayers(rasterBrickSpectralVNIR)){
    y1 <-  subset(rasterBrickSpectralVNIR, (wavelengthVNIROrderNum))
    y2 <-  subset(rasterBrickSpectralVNIR, (wavelengthVNIROrderNumPlusOne))
   #} else break
  }
  # apply the linear interpolation equation
  # the variables for the linear interpolation are explained as follows
  # x = desired wavelength
  # x1 = wavelength just less than of desired wavelength
  # x2 = wavelength just greater than of desired wavelength
  # y = desired values at the required wavelength
  # y1 = band data at corresponding x1 location
  # y2 = band data at corresponding x2 location
  x <- i
  #cat ("\nInterpolating wavelength :", i)
  x1 <- as.numeric(x1)
  x2 <- as.numeric(x2)
  x3 = (x - x1)
  y3 = (y2 - y1)
  x4 = (x2 + x1)
  xy1 = x3 * y3
  xy2 = xy1 / x4
  y = y1 + xy2
  y <- y1 + ((x - x1)*((y2 - y1)/(x2 - x1)))
  temp5 = y
 }
 #cat ("\nStacking Band:", i)
   newRasterBrickVNIR <- stack(newRasterBrickVNIR, y) # stack each of the desired wavelength bands
}
# remove the first band as it was temporarily necessary during the stack creation
```

```
newRasterBrickVNIR <- dropLayer(newRasterBrickVNIR, 1)

#output the new super spectral resolution image
outputfilename <- "SuperSpectralImageVNIR.tif"
writeRaster(newRasterBrickVNIR, filename=(outputfilename), overwrite=TRUE)
print ("Done creating a 1 nm spectral resolution image for gaussian weighting. Now applying gaussian weighting...")

# set the standard devation for the FWHM using the desired spectral resolution as input
standardDev = desiredSpectralResVNIR/2.3548
if (desiredSpectralResVNIR == 6.5){
  x <- seq(-10, 10, length=21) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
if (desiredSpectralResVNIR == 7.5){
  x <- seq(-12, 12, length=25) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
if (desiredSpectralResVNIR == 10){
  x <- seq(-15, 15, length=31) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
f <- function(x) {
  y <- 5 *(1/(sqrt(2*pi*standardDev)))*(2.718281828459^((- ((x^2) / (2*(standardDev^2))))))  # 1 is the vertical scale
}

SRF = f (x) #create the weighted values in the spectral response from each 1 nm interval
SRF_Sum = (SRF)/sum(SRF) #have the values sum to one.
SRF_normalized <- (SRF - (min(SRF)))/(max(SRF)-min(SRF)) # not necessary but good for confirming the FWHM value is
# correct

# In the case where the desired resolution is 'not' a whole number the guassian weighting
# calculated above will not work. This is because the peak values for the gaussian wieghting
# in vectors: x & SRF_Sum fall squarely on the central vector. With desired resolutions that
# contains half nm this isn't the case. So new vectors (xHalfNumber & SRF_SumHalfNumber) have
# to be created.

if (desiredSpectralResVNIR == 6.5){
  xHalfNumber <- seq(-10, 10, length=20)  # length is changing the sampling interval. these values assume the input
  # has a spectral resolution of 1 nm
  SRFHalfNumber = f (xHalfNumber) #create the weighted values in the spectral response from each 1 nm interval
  SRF_SumHalfNumber = (SRFHalfNumber)/sum(SRFHalfNumber)} #have the values sum to one.

if (desiredSpectralResVNIR == 7.5){
  xHalfNumber <- seq(-12, 12, length=25)  # length is changing the sampling interval. these values assume the input
  # has a spectral resolution of 1 nm
  SRFHalfNumber = f (x) #create the weighted values in the spectral response from each 1 nm interval
  SRF_SumHalfNumber = (SRFHalfNumber)/sum(SRFHalfNumber)} #have the values sum to one.

################################################################################################
# Now the script will iteratively convolve the high spectral resolution to a lower resolution.
# It does this by iteratively weighting the necessary bands across the complete spectra.

nBandIterationsVNIR <- (VNIRmaximumWavelength - VNIRminimumWavelengthBuff)/desiredSpectralResVNIR
nBandIterationsVNIR = floor(nBandIterationsVNIR) # round the number of layers divided by the desired spectral
# resolution
nBandIterationsVNIR = 1:(nBandIterationsVNIR) # convert the number of iterations to a vector list starting from 1
bandPosition = nBandIterationsVNIR * desiredSpectralResVNIR # create a vector of the spectral resolution positions
SRFOrder = 1:length(x) #create a list of the order for the weighting to be applied later.
#SRFOrderHalfNumber = 1:length(xHalfNumber) #create a list of the order for the weighting to be applied later.
nBandIterationsVNIRx10 <- nBandIterationsVNIR * desiredSpectralResVNIR
nBandIterationsVNIRx10 <- nBandIterationsVNIRx10 - desiredSpectralResVNIR
newRasterBrick3 <- subset (newRasterBrickVNIR, 1) # temporary rasterBrick to be used later

for (t in (nBandIterationsVNIR)){ # repeat the following n number of times. n is the number of 10, 6.5, or 7.5 nm
  # bands that fit into the spectral range of the input data
  newRasterBrick2 <- subset (newRasterBrickVNIR, 1)# temporary rasterBrick to be used later
  #print (paste("iteration" ,  (t))) #print to screen the band the script is processing
  for (n in (SRFOrder)){
    temp0 <- n + nBandIterationsVNIRx10 [t] #creates a variable of the current band number being processed. This is
    # SRF order number + wavelength interval (ie. 6.5, 13, 19.5,...)
    #print (temp0)
    temp9 = nlayers(newRasterBrickVNIR) + 1 #create a variable of the highest number of bands that can be processed
    # plus 1. this variable remains fixed through the following sections
    #print (temp9)
    temp5 <- temp0 + desiredSpectralResVNIR
    if (temp0 == temp9) {break} # stop looping when the spectral range has been processed
    if (temp0 > temp9) {break} # stop looping when the spectral range has been processed. Greater than is used for
    # when the spectral resolution is a half humber

    ###################################
    #create a function to test if a number is a whole number (in this case we are testing if the desired wavelength
    # is a whole number)
    is.wholenumber <- function(temp0, tol = .Machine$double.eps^0.5)  abs(temp0 - round(temp0)) < tol
```

```
  is.wholenumber(1) # is TRUE
  ####################################
  if (is.wholenumber( temp0 ) ){ # if the desired wavelength is a whole number then use appropriate a band weighting
    temp10 <- subset (newRasterBrickVNIR, (temp0)) #subset the band depending on the position of the band for
    # gaussian weighting
    temp2 <- SRF_Sum [(n)] #create a variable of the gaussian weighting depending on the position of the band
  } else { # if the desired wavelength is not a whole number then use appropriate band weighting
    temp10 <- subset (newRasterBrickVNIR, (temp0)) #subset the band depending on the position of the band for
    # gaussian weighting
    temp2 <- SRF_SumHalfNumber [(n)]]#create a variable of the gaussian weighting depending on the position of the
  # band
  temp3 <- temp10 * temp2 #apply, in this case multiply, the weighting (temp2) on the appropriate band (temp2)
  newRasterBrick2 <- addLayer(newRasterBrick2, temp3) #stack each iteration of the band weighting so that it can be
  # sumed later
  #print (paste("SRF band" ,  (n))) #print to screen which band is being processed
  numLayers <- nlayers (newRasterBrick2) #create a variable of the number of bands in the gaussian weighting so the
  # script can stop when the correct number have been processed
   if (desiredSpectralResVNIR == 6.5){
    if (numLayers > 21) {break}
  }
  if (desiredSpectralResVNIR == 7.5){
    if (numLayers > 24) {break}
  }
  if (desiredSpectralResVNIR == 10){
    if (numLayers > 31) {break}
  }
 }

 if (temp0 == temp9) {break} # stop looping when the spectral range has been processed
 if (temp0 > temp9) {break} # stop looping when the spectral range has been processed. Greater than is used for when
 # the spectral resolution is a half humber

 newRasterBrick2 <- dropLayer(newRasterBrick2, 1)
 if (desiredSpectralResVNIR == 6.5){
  if (is.wholenumber( temp0 ) ){
    newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
    subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) + subset(newRasterBrick2, 7)
    + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) + subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) + subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) +
subset(newRasterBrick2, 18) + subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21)
  }
  if (is.wholenumber( temp0 ) + 0.5 ){
    newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
    subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) + subset(newRasterBrick2, 7)
    + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) + subset(newRasterBrick2, 10) +
      subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) + subset(newRasterBrick2, 13) +
      subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) + subset(newRasterBrick2, 16) +
      subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) + subset(newRasterBrick2, 19) +
      subset(newRasterBrick2, 20)
  }
 }
 if (desiredSpectralResVNIR == 7.5){
  if (is.wholenumber( temp0 ) ){
    newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
      subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23) + subset(newRasterBrick2, 24)
  }
  if (is.wholenumber( temp0 ) + 0.5 ){
    newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
      subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23)
  }
 }
 if (desiredSpectralResVNIR == 10){
   newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
    subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
    subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
    subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
    subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
    subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
```

```
    subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
    subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23) + subset(newRasterBrick2, 24) +
    subset(newRasterBrick2, 25) + subset(newRasterBrick2, 26) + subset(newRasterBrick2, 27) +
    subset(newRasterBrick2, 28) + subset(newRasterBrick2, 29) + subset(newRasterBrick2, 30) +
    subset(newRasterBrick2, 31)
 }
 newRasterBrick3 <- addLayer (newRasterBrick3, newRaster1)
}
spectralConvRasterVNIR <- dropLayer (newRasterBrick3, 1) # drop the first layer(it was a temporary layer when the
# brick was created earlier) and retain the convolved layers

# Create a vector object of the wavelength information for copying into the output header file
wavelengthMin <- min(desiredWavelengthWhole5)  + (VNIRBuffer)
wavelengthnLayers <- nlayers (spectralConvRasterVNIR)
wavelengthMax <- wavelengthMin + (wavelengthnLayers * desiredSpectralResVNIR) - desiredSpectralResVNIR
wavelengthVNIROut <-  seq(from=wavelengthMin, to=wavelengthMax+desiredSpectralResVNIR, by=desiredSpectralResVNIR)
wavelengthVNIROut <-  wavelengthVNIROut + 1
length (wavelengthVNIROut)
if (length (wavelengthVNIROut) > nlayers (spectralConvRasterVNIR)){
 wavelengthVNIROut <- head(wavelengthVNIROut, -1)
}

# Create a vector object of the bandwidth information for copying into the output header file
fwhm = desiredSpectralResVNIR * .001
numlayers <- nlayers (spectralConvRasterVNIR)
fwhmrepVNIR = rep(fwhm, numlayers)

# Change pixel values in the output raster brick stored as 0s to NAs
rasterBrickSpectralVNIR[0] <- NA

# Plot the data as a visual check
#plotRGB (rasterBrickSpectralVNIR, 2, 4, 1, stretch = 'lin')
print ("Done Processing VNIR Sensor information. Now processing SWIR sensor...")

##############################################################################################
##############################################################################################
#SWIR sensor
##############################################################################################
##############################################################################################
# Find the closest wavelength in the new 1 nm vector list to just below the desired minimum spectral range value
# (900nm for EnMap) while considering the buffer required for gaussian weighting of the input bands. This value is
# 15 nm for a spectral resolution of 10 nm. In other words there is an additional value of spectral resolution
# required in order to convolve the bands to the desired resolution.The amount of buffer is controlled by the shape
# of the guassian sampling distance.

if (desiredSpectralResSWIR == 6.5){
 SWIRBuffer <- 10}
if (desiredSpectralResSWIR == 7.5){
 SWIRBuffer <- 12}
if (desiredSpectralResSWIR == 10){
 SWIRBuffer <- 15}

SWIRminimumWavelengthBackUp <- SWIRminimumWavelength
SWIRminimumWavelength <- SWIRminimumWavelength - (SWIRBuffer + 1) #resetting the SWIR minimum wavelength vector list
# to consider the necessary buffer
# We now need to create a table to be used later. The table will be used to create vectors of the band order and the
# associated next wavelength
for (i in (SWIRminimumWavelength)){
 dt <- data.table(wavelength, val = wavelength) # create a data table of wavelength
 setattr(dt, "sorted", "wavelength")
 # write a function (called DFshift) that will allow the values in a data table to populate a column with either the
 # following or preceeding values in a table
 DFshift <- function(x, offset = 1, pad = NA) {
  r <- (1 + offset):(length(x) + offset)
  r[r<1] <- NA
  ans <- x[r]
  ans[is.na(ans)] <- pad
  return(ans)
 }
 dt[, D := DFshift(val,-1)]#add a column that contains all the values of the preceeding row. This is necessary for
 # the steps below.
 temp1 <- dt[J(i), roll = "nearest"]
 wavelengthOrderNumMin <-sapply(i,function(i)which.min(abs(i - wavelength)))
 closestWavelength <-subset(temp1, select=c(val))
 # change the record number if the closest record is greater than the minimum required
 # wavelength
 if(closestWavelength > i){
  closestWavelength <-subset(temp1, select=c(D))
  #use the value from the row above. This is actually stored in the column "D"
 } # change the val (order number) to one record lower than the closest value so that now the
```

```
  #subset will be the closest record but lower than the minimum value.
}

# find the closest wavelength in the new 1 nm vector file to just above the maximum desired spectral range value
# (1000nm for EnMap's SWIR) while considering the buffer required to gaussian weighting of the input bands. In
# other words there is an additional value of spectral resolution required in order for the convolvion of the bands.
SWIRmaximumWavelength <- SWIRmaximumWavelength + SWIRBuffer
for (i in (SWIRmaximumWavelength)){
  dt2 <- data.table(wavelength, val = wavelength)
  setattr(dt2, "sorted", "wavelength")
  dt2[, D := DFshift(val,+1)]#add a column that contains all the values of the preceeding row. This is necessary for
  # the steps below.
  temp1 <- dt2[J(i), roll = "nearest"]
  wavelengthOrderNumMax <-sapply(i,function(i)which.min(abs(i - wavelength)))
  closestMaxWavelength <-subset(temp1, select=c(val))
  #change therecord number if the closest record greater than the minimum required EnMap wavelegnth
  #closestMaxWavelength = 904
  if(closestMaxWavelength < i){
    closestMaxWavelength <-subset(temp1, select=c(D)) #use the value from the row above. This is actually stored in the
    # column "D"
  } # change the val (order number) to one record lower than the closest value so that now the subset will be the
  # closest record but lower than the minimum value.
}

# maxBandfromInputFileNum <- (dt2[[1]])
# maxBandfromInputFile <- max(dt2[[1]])
# maxBandfromInputFile <- which (maxBandfromInputFileNum==maxBandfromInputFile)

# subset the SWIR bands from the input data
nlayersMinus1 <- wavelengthOrderNumMin - 1
if (nlayersMinus1 < 0){
  nlayersMinus1 <- 1
}

# drop the lower bands than those needed for processing.
rasterBrickSpectralSWIR <- dropLayer(rBrickSpectral1,c(0:nlayersMinus1))

#Create a vector containing the corresponding wavelengths for the SWIR raster Brick bands of the desired output
# Strip the last records from the wavelength vector
wavelengthSWIR <- head(wavelength, (wavelengthOrderNumMax)) # for stripping the last records from the wavelength vector
# Strip the first records from the wavelength vector
wavelengthLength <- length(wavelengthSWIR) - wavelengthOrderNumMin + 1
wavelengthSWIR <- tail(wavelengthSWIR, (wavelengthLength)) # for stripping the first records from the wavelength vector

#############################################################################################
# before interpolating strip out the first bands below the minimum spectral range value
# determined at the start of this script. This was user defined
desiredWavelengthWhole5 <- desiredWavelengthWhole5backup
#desiredWavelengthWhole5backup <- desiredWavelengthWhole5
while (min(desiredWavelengthWhole5) < SWIRminimumWavelength) {
  desiredWavelengthWhole5 <- tail(desiredWavelengthWhole5, length(desiredWavelengthWhole5) - 1)}
while (max(desiredWavelengthWhole5) > SWIRmaximumWavelength) {
  desiredWavelengthWhole5 <- head(desiredWavelengthWhole5, length(desiredWavelengthWhole5) - 1)}
desiredWavelengthWhole6 <- min(desiredWavelengthWhole5) - 1
desiredWavelengthWhole5 <- append (desiredWavelengthWhole6, desiredWavelengthWhole5)
#############################################################################################
#linearly interpolate each desired wavelength to create 1 nm super resolution image
for (i in (desiredWavelengthWhole5)){
  # find the nearest wavelength to the desired wavelength and determine which record this is
  # in the R vector
  dt3 <- data.table(wavelengthSWIR, val = wavelengthSWIR)
  setattr(dt3, "sorted", "wavelengthSWIR")
  temp1 <- dt3[J(i), roll = "nearest"]
  #print (temp1)
  wavelengthSWIROrderNum <-sapply(i,function(i)which.min(abs(i - wavelengthSWIR)))
  closestWavelength <-subset(temp1, select=c(val))

  # determine the X values for the interpolation equation
  if(closestWavelength < i){
    x1 <-  closestWavelength
    wavelengthSWIROrderNumPlusOne <- (wavelengthSWIROrderNum) + 1
    x2 <-  wavelengthSWIR[(wavelengthSWIROrderNumPlusOne)]

  } else if(closestWavelength == i){
    x1 <-  closestWavelength
    wavelengthSWIROrderNumPlusOne <- (wavelengthSWIROrderNum) + 1
    x2 <-  wavelengthSWIR[(wavelengthSWIROrderNumPlusOne)]

  } else { # when the closest wavelength is greater than the desired wavelength
    wavelengthSWIROrderNumMinusOne <- (wavelengthSWIROrderNum) - 1
```

```
  if (wavelengthSWIROrderNum == 1){
    next
  }
  x1 <- wavelengthSWIR[(wavelengthSWIROrderNumMinusOne)]
  x2 <- closestWavelength
 }

 # determine the y values for the interpolation equation
 if(i == min(wavelengthSWIR)){
  y <- subset (rasterBrickSpectralSWIR,1)
 } else if(i == max(wavelengthSWIR)){
  y <- nlayers(rasterBrickSpectralSWIR)
  y <- subset (rasterBrickSpectralSWIR,(y))
 } else if(i == closestWavelength){
  y <- subset (rasterBrickSpectralSWIR,(wavelengthSWIROrderNum))

 } else {   #interpolate the remaining desired wavelengths
  # apply the linear interpolation equation but first set the remaining variables
  x <- i
  if(closestWavelength > i){      # when the closest wavelength is greater than the desired wavelength
    wavelengthSWIROrderNumMinusOne <- (wavelengthSWIROrderNum) - 1
    y1 <-  subset(rasterBrickSpectralSWIR, (wavelengthSWIROrderNumMinusOne))
    y2 <-  subset(rasterBrickSpectralSWIR, (wavelengthSWIROrderNum))

  } else {  # when the closest wavelength is less than the desired wavelength
    wavelengthSWIROrderNumPlusOne <- (wavelengthSWIROrderNum) + 1
    if (nlayers(rasterBrickSpectralSWIR) >= wavelengthSWIROrderNumPlusOne){
      y1 <-  subset(rasterBrickSpectralSWIR, (wavelengthSWIROrderNum))
      y2 <-  subset(rasterBrickSpectralSWIR, (wavelengthSWIROrderNumPlusOne))
    } else break
  }
  # apply the linear interpolation equation
  # the variables for the linear interpolation are explained as follows
  # x = desired wavelength
  # x1 = wavelength just less than of desired wavelength
  # x2 = wavelength just greater than of desired wavelength
  # y = desired values at the required wavelength
  # y1 = band data at corresponding x1 location
  # y2 = band data at corresponding x2 location
  x <- i
  #print (x)
  x1 <- as.numeric(x1)
  x2 <- as.numeric(x2)
  x3 = (x - x1)
  y3 = (y2 - y1)
  x4 = (x2 + x1)
  xy1 = x3 * y3
  xy2 = xy1 / x4
  y = y1 + xy2
  y <- y1 + ((x - x1)*((y2 - y1)/(x2 - x1)))
  temp5 = y
 }
 newRasterBrick <- stack(newRasterBrick, y) # stack each of the desired wavelength bands
}
newRasterBrick <- dropLayer(newRasterBrick, 1) #remove the first band as it was temporarily necessary during the
# stack creation

newRasterBrickSWIR <- newRasterBrick

##output the new super spectral resolution image
outputfilename <- "SuperSpectralImageSWIR.tif"
writeRaster(newRasterBrick, filename=(outputfilename), overwrite=TRUE)

print ("Done creating a 1 nm spectral resolution image for gaussian weighting. Now applying gaussian weighting...")

# set the standard devation for the FWHM using the desired spectral resolution as input
standardDev = desiredSpectralResSWIR/2.3548 # set the standard devation for the FWHM using the desired spectral
# resolution as input
if (desiredSpectralResSWIR == 6.5){
 x <- seq(-10, 10, length=21) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
if (desiredSpectralResSWIR == 7.5){
 x <- seq(-12, 12, length=25) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
if (desiredSpectralResSWIR == 10){
 x <- seq(-15, 15, length=31) } # length is changing the sampling interval. these values assume the input has a
# spectral resolution of 1 nm
f <- function(x) {
 y <- 5 *(1/(sqrt(2*pi*standardDev)))*(2.718281828459^((- ((x^2) / (2*(standardDev^2))))))  # 1 is the vertical scale
}
```

```
SRF = f (x) #create the weighted values in the spectral response from each 1 nm interval
SRF_Sum = (SRF)/sum(SRF) #have the values sum to one.
SRF_normalized <- (SRF - (min(SRF)))/(max(SRF)-min(SRF)) # not necessary but good for confirming the FWHM value is
# correct

# In the case where the desired resolution is 'not' a whole number the guassian weighting
# calculated above will not work. This is because the peak values for the gaussian wieghting
# in vectors: x & SRF_Sum fall squarely on the central vector. With desired resolutions that
# contains half nm this isn't the case. So new vectors (xHalfNumber & SRF_SumHalfNumber) have
# to be created.

if (desiredSpectralResSWIR == 6.5){
  xHalfNumber <- seq(-10, 10, length=20)  # length is changing the sampling interval. these values assume the input
  # has a spectral resolution of 1 nm
  SRFHalfNumber = f (xHalfNumber) #create the weighted values in the spectral response from each 1 nm interval
  SRF_SumHalfNumber = (SRFHalfNumber)/sum(SRFHalfNumber)} #have the values sum to one.

if (desiredSpectralResSWIR == 7.5){
  xHalfNumber <- seq(-12, 12, length=25)  # length is changing the sampling interval. these values assume the input
  # has a spectral resolution of 1 nm
  SRFHalfNumber = f (x) #create the weighted values in the spectral response from each 1 nm interval
  SRF_SumHalfNumber = (SRFHalfNumber)/sum(SRFHalfNumber)} #have the values sum to one.

# ##############################################################################################
# # The following section is for making plots of the spectral sampling weighting for when the
# # desired resolution is a whole number.
# if (desiredSpectralResSWIR == 6.5){
#   x2 <- seq(-3.5, 16.5, length=21)
#   x3 <- seq(3, 23, length=21)
#   x4 <- seq(-16.5, 3.5, length=21)
#   x5 <- seq(-23, -3, length=21)
# }
# if (desiredSpectralResSWIR == 7.5){
#   x2 <- seq(-4.5, 19.5, length=25)
#   x3 <- seq(3, 27, length=25)
#   x4 <- seq(-19.5, 4.5, length=25)
#   x5 <- seq(-27, -3, length=25)
# }
# if (desiredSpectralResSWIR == 10){
#   x2 <- seq(-5, 25, length=31)
#   x3 <- seq(5, 35, length=31)
#   x4 <- seq(15, 45, length=31)
#   x5 <- seq(-25, 5, length=31)
#   x6 <- seq(-35, -5, length=31)
# }
#
# plot(x, SRF_Sum, type="b", xlab="nm", ylab="Spectral Response", ylim=c(0,max(SRF_Sum))) #when the desired resoltuion is 7.5 nm
# #plot(x, SRF_Sum, type="b", xlab="nm", ylab="Spectral Response", ylim=c(0,.05) ) #when the desired resoltuion is 10 nm
# lines (x2, SRF_Sum, lwd = 1.0, col = "grey")
# length (x3)
# length (SRF_Sum)
# lines (x3, SRF_Sum, lwd = 1.0, col = "grey")
# lines (x4, SRF_Sum, lwd = 1.0, col = "grey")
# lines (x5, SRF_Sum, lwd = 1.0, col = "grey")
# ##############################################################################################
# #This section is for making plots of the spectral sampling weighting for when the desired resolution is a half number
# if (desiredSpectralResSWIR == 6.5){
#   standardDev = desiredSpectralResSWIR/2.3548 # set the standard devation for the FWHM using the desired spectral resolution as input
#   xHalfNumber <- seq(-10, 10, length=20)  # length is changing the sampling interval. these values assume the input has a spectral resolution of 1 nm
#   xHalfNumber2 <- seq(-3.5, 16.5, length=20)
#   xHalfNumber3 <- seq(3, 23, length=20)
#   xHalfNumber4 <- seq(-16.5, 3.5, length=20)
#   xHalfNumber5 <- seq(-23, -3, length=20)
#   plot(xHalfNumber, SRF_SumHalfNumber, type="b", xlab="nm", ylab="Spectral Response", ylim=c(0,max(SRF_Sum))) #when the desired resoltuion is 7.5 nm
#   lines (  xHalfNumber2, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber3, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber4, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber5, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
# }
# if (desiredSpectralResSWIR == 7.5){
#   x2 <- seq(-4.5, 19.5, length=24)
#   x3 <- seq(3, 27, length=24)
#   x4 <- seq(-19.5, 4.5, length=24)
#   x5 <- seq(-27, -3, length=24)
#   plot(xHalfNumber, SRF_SumHalfNumber, type="b", xlab="nm", ylab="Spectral Response", ylim=c(0,max(SRF_Sum))) #when the desired resoltuion is 7.5 nm
#   lines (  xHalfNumber2, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber3, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber4, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
#   lines (  xHalfNumber5, SRF_SumHalfNumber, lwd = 1.0, col = "grey")
```

```
# }

#############################################################################################
# Now the script will iteratively convolve the high spectral resolution to a lower resolution.
# It does this by iteratively weighting the necessary bands across the complete spectra.
# First round down the number of iterations required to a full number. This is the number of
# output bands
if ((SWIRmaximumWavelength-desiredSpectralResSWIR) > tail(wavelength,1)){
  nBandIterationsSWIR <- (tail(wavelength,1) -  SWIRminimumWavelength)/desiredSpectralResSWIR
  } else {
    nBandIterationsSWIR <- ((SWIRmaximumWavelength-desiredSpectralResSWIR)-
                        (SWIRminimumWavelength+desiredSpectralResSWIR))/desiredSpectralResSWIR
  }
nBandIterationsSWIR = floor(nBandIterationsSWIR) # round the number of layers divided by the desired spectral
# resolution
nBandIterationsSWIR1 <- nBandIterationsSWIR
nBandIterationsSWIR = 1:(nBandIterationsSWIR) # convert the number of iterations to a vector list starting from 1
bandPosition = nBandIterationsSWIR * desiredSpectralResSWIR # create a vector of the spectral resolution positions
SRFOrder = 1:length(x) #create a list of the order for the weighting to be applied later.
#SRFOrderHalfNumber = 1:length(xHalfNumber) #create a list of the order for the weighting to be applied later.
nBandIterationsSWIRx10 <- nBandIterationsSWIR * desiredSpectralResSWIR
nBandIterationsSWIRx10 <- nBandIterationsSWIRx10 - desiredSpectralResSWIR
newRasterBrick3 <- subset (newRasterBrick, 1) # temporary rasterBrick to be used later

for (t in (nBandIterationsSWIR)){ # repeat the following n number of times. n is the number of 10, 6.5, or 7.5 nm
  # bands that fit into the spectral range of the input data
  newRasterBrick2 <- subset (newRasterBrick, 1)# temporary rasterBrick to be used later
  #print (paste("iteration" ,  (t))) #print to screen the band the script is processing
  for (n in (SRFOrder)){
    temp0 <- n + nBandIterationsSWIRx10 [t] #creates a variable of the current band number being processed. This is
    # SRF order number + wavelength interval (ie. 6.5, 13, 19.5,...)
    #print (temp0)
    temp9 = nlayers(newRasterBrick) + 1 #create a variable of the highest number of bands that can be processed plus 1.
    # this variable remains fixed through the following sections
    #print (temp9)
    if (temp0 == temp9) {break} # stop looping when the spectral range has been processed
    if (temp0 > temp9) {break} # stop looping when the spectral range has been processed. Greater than is used for when
    # the spectral resolution is a half humber

    ####################################
    #create a function to test if a number is a whole number (in this case we are testing if the desired wavelength is a
    # whole number)
    is.wholenumber <- function(temp0, tol = .Machine$double.eps^0.5)  abs(temp0 - round(temp0)) < tol
    is.wholenumber(1) # is TRUE
    ####################################
    if (is.wholenumber( temp0 ) ){ # if the desired wavelength is a whole number then use appropriate a band weighting
      temp1 <- subset (newRasterBrick, (temp0) ) #subset the band depending on the position of the band for gaussian
      # weighting
      temp2 <- SRF_Sum [(n)] #create a variable of the gaussian weighting depending on the position of the band
    } else { # if the desired wavelength is not a whole number then use appropriate band weighting
      temp1 <- subset (newRasterBrick, (temp0) ) #subset the band depending on the position of the band for gaussian
      # weighting
      temp2 <- SRF_SumHalfNumber [(n)]]#create a variable of the gaussian weighting depending on the position of the band
    temp3 <- temp1 * temp2 #apply, in this case multiply, the weighting (temp2) on the appropriate band (temp2)
    newRasterBrick2 <- addLayer(newRasterBrick2, temp3) #stack each iteration of the band weighting so that it can be
    # sumed later
    #print (paste("SRF band" ,  (n))) #print to screen which band is being processed
    numLayers <- nlayers (newRasterBrick2) #create a variable of the number of bands in the gaussian weighting so the
    # script can stop when the correct number have been processed
      if (desiredSpectralResSWIR == 6.5){
      if (numLayers > 21) {break}
    }
    if (desiredSpectralResSWIR == 7.5){
      if (numLayers > 24) {break}
    }
    if (desiredSpectralResSWIR == 10){
      if (numLayers > 31) {break}
    }
  }

  if (temp0 == temp9) {break} # stop looping when the spectral range has been processed
  if (temp0 > temp9) {break} # stop looping when the spectral range has been processed. Greater than is used for when
  # the spectral resolution is a half humber

  newRasterBrick2 <- dropLayer(newRasterBrick2, 1)
  if (desiredSpectralResSWIR == 6.5){
    if (is.wholenumber( temp0 ) ){
      newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
        subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
        subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
```

```
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21)
   }
   if (is.wholenumber( temp0 ) + 0.5 ){
     newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20)
   }
  }
  if (desiredSpectralResSWIR == 7.5){
   if (is.wholenumber( temp0 ) ){
     newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
      subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23) + subset(newRasterBrick2, 24)
   }
   if (is.wholenumber( temp0 ) + 0.5 ){
     newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
      subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23)
   }
  }
  if (desiredSpectralResSWIR == 10){
    newRaster1 <- subset(newRasterBrick2, 1) + subset(newRasterBrick2, 2) + subset(newRasterBrick2, 3) +
      subset(newRasterBrick2, 4) + subset(newRasterBrick2, 5) + subset(newRasterBrick2, 6) +
      subset(newRasterBrick2, 7) + subset(newRasterBrick2, 8) + subset(newRasterBrick2, 9) +
      subset(newRasterBrick2, 10) + subset(newRasterBrick2, 11) + subset(newRasterBrick2, 12) +
      subset(newRasterBrick2, 13) + subset(newRasterBrick2, 14) + subset(newRasterBrick2, 15) +
      subset(newRasterBrick2, 16) + subset(newRasterBrick2, 17) + subset(newRasterBrick2, 18) +
      subset(newRasterBrick2, 19) + subset(newRasterBrick2, 20) + subset(newRasterBrick2, 21) +
      subset(newRasterBrick2, 22) + subset(newRasterBrick2, 23) + subset(newRasterBrick2, 24) +
      subset(newRasterBrick2, 25) + subset(newRasterBrick2, 26) + subset(newRasterBrick2, 27) +
      subset(newRasterBrick2, 28) + subset(newRasterBrick2, 29) + subset(newRasterBrick2, 30) +
      subset(newRasterBrick2, 31)
  }
  newRasterBrick3 <- addLayer (newRasterBrick3, newRaster1)
}
spectralConvRasterSWIR <- dropLayer (newRasterBrick3, 1) # drop the first layer(it was a temporary layer when the brick
# was created earlier) and retain the convolved layers

#############################################################################################
# Create a vector object of the wavelength information for copying into the output header file
wavelengthMin <- min(desiredWavelengthWhole5) + 1 + (SWIRBuffer)
wavelengthnLayers <- nlayers (spectralConvRasterSWIR)
wavelengthMax <- wavelengthMin + (wavelengthnLayers * desiredSpectralResSWIR)
wavelengthMin <- wavelengthMin +1
wavelengthSWIROut <-  seq(from=wavelengthMin, to=wavelengthMax, by=desiredSpectralResSWIR)
if (length (wavelengthSWIROut) > nlayers (spectralConvRasterSWIR)){
  difference <- length (wavelengthSWIROut) - (nlayers (spectralConvRasterSWIR))
  wavelengthSWIROut <- head(wavelengthSWIROut, -(difference))
}
#due to the shoulders of the gaussian SRF the highest wavelengths may not get produced. Here these bands are simulated
# using the interpolatd values rather than using a gaussian SR if data exists
SWIRWavelengths <- nBandIterationsSWIRx10 + SWIRminimumWavelengthBackUp
# check if all the necessary bands are simulated
if (max(wavelengthSWIROut) < max(SWIRWavelengths)){
  #discover which bands are missing
  missingBands <- setdiff(SWIRWavelengths,wavelengthSWIROut)
  #now add the missing bands form the interpolated raster brick
  for (i in (missingBands)) {
    if (max(SWIRWavelengths) > i) { #only accomplish this if there is a band in the interpolated rasterBrick
      missingBandNum <- which (desiredWavelengthWhole5==i)
      bandToAdd <- subset (newRasterBrick,(missingBandNum))
      spectralConvRasterSWIR <- addLayer(spectralConvRasterSWIR, bandToAdd)
      wavelengthSWIROut <- append(wavelengthSWIROut, i)
```

```
   } else # otherwise use the next highest band from the input data (non-interpolated).
     if (i < max(SWIRWavelengths)){
       bandToAdd <- subset (rBrickSpectral1,nlayers (rBrickSpectral1))
       spectralConvRasterSWIR <- addLayer(spectralConvRasterSWIR, bandToAdd)
       wavelengthSWIROut <- append(wavelengthSWIROut, i)
     }
  }
}

# Create a vector object of the bandwidth information for copying into the output header file
fwhm = desiredSpectralResSWIR * .001
numlayers <- nlayers (spectralConvRasterSWIR)
fwhmrepSWIR = rep(fwhm, numlayers)

#Change the 0 values in the output images to NAs
rasterBrickSpectralSWIR[0] <- NA


# Remove the extra band wavelength created at the beginning of the script. This is created to overcome the error
# caused by the last layer
SWIRmaximumWavelength <- SWIRmaximumWavelength - desiredSpectralResSWIR

##############################################################################################
#Final Report
print ("Report on output spectral convolution:")
print ("The user now needs to ensure the band information is copied to the output image header file ")
print ("Open the VNIR_band_Info.csv in excel and delete the first coloumn and first row (column headings)")
print  ("Save and close the file")
print ("Then in ENVI edit the output imagery header (metadata) file by importing the file just saved")
temp <- length(wavelengthVNIROut)
fwhm <- head(fwhmrepVNIR,1)
fwhm <- rep(fwhm, len = temp) #create a vector with the FWHM value for each band
dfVNIR <- data.frame(as.table(setNames(fwhm, wavelengthVNIROut)))
write.table(dfVNIR, file = "VNIR_band_Info.csv", sep = ",", col.names = NA, qmethod = "double")

print ("Open the SWIR_band_Info.csv in excel and delete the first coloumn and first row (column headings).")
print  ("Save and close the file")
print ("Then in ENVI edit the output imagery header (metadata) file by importing the file just saved")
temp <- length(wavelengthSWIROut)
fwhm <- head(fwhmrepSWIR,1)
fwhm <- rep(fwhm, len = temp) #create a vector with the FWHM value for each band
dfSWIR <- data.frame(as.table(setNames(fwhm, wavelengthSWIROut)))
write.table(dfSWIR, file = "SWIR_band_Info.csv", sep = ",", col.names = NA, qmethod = "double")

##############################################################################################
#output the spectral convolution image
outputfilename <- "SpectralResampleDataOutSWIR.tif" # * change the name as needed
outputSWIR <- writeRaster(spectralConvRasterSWIR, filename=(outputfilename), overwrite=TRUE)

#output the spectral convolution image
outputfilename <- "SpectralResampleDataOutVNIR4.tif" # * change the name as needed
outputVNIR <- writeRaster(spectralConvRasterVNIR, filename=(outputfilename), overwrite=TRUE, datatype='INT4S',
                          bandorder='BIL')

#Create a single image of all bands but NOT removing the overlapping bands in the SWIR
spectralConvRaster <- stack (spectralConvRasterVNIR, spectralConvRasterSWIR)

# create a vector with the wavelength position information
#first add 0.0001 to the SWIR values to make them uniquely different if the same value is in the VNIR bands.
wavelengthSWIROut <- wavelengthSWIROut + 0.001
outputWavelengthPositions <- append(wavelengthVNIROut, wavelengthSWIROut)

#stop the stopwatch to see how long the script took to run
#proc.time() - starttime
#print (proc.time() - starttime)
# end of script
```

```
##############################################################################################
# 4. Noise Modelling.R
# Script details:
# This script adds signal dependent and independent (calibration, thermal, read, flicker)
# noise to input hyperspectral data. The script outputs three datacubes including the
# simulated scene with any overlapping bands, another simulated scene without overlapping
# bands (priority given to the VNIR bands), and finally a total added noise datacube. Also
# included are .CSV files contianing the band wavelength and FWHM information for the
# simulated datacubes.
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada -
# Natural Resources Canada
#
# Script completed December, 2016
##############################################################################################
#start a stopwatch to time the script
#starttime <- proc.time()
# Load the input dataset
rBrickInput <- spectralConvRaster
##############################################################################################
#Set all values that should be NA
rBrickInput [is.nan(rBrickInput)] = NA
rBrickInput [rBrickInput==(NAValues)] <-NA
rBrickInput [rBrickInput==-2147483647] <-NA
rBrickInput [rBrickInput==-1.7e+308] <-NA
rBrickInput [rBrickInput<0] <-NA
rBrickInput [is.infinite(rBrickInput)] = NA
rBrickInputMask <- rBrickInput
#rBrickInput <- extend(rBrickInput, c(3,3))
#rBrickInput [is.na(rBrickInput)] = noiseReflectance # Change all the NA values to the value of reflectanace needed for
# the SNR calculation
rBrickNoise <- subset (rBrickInput, 1) # prepare to have an output raster with the appropriate extents,
# projection, and resolution by copying a band from this raster brick
# Obtain the dimensions of hte raster brick
matrixNrow <- nrow (rBrickInput)
matrixNcol <- ncol (rBrickInput)
rasterXmin <- xmin(rBrickInput)
rasterXmax <- xmax(rBrickInput)
rasterYmin <- ymin(rBrickInput)
rasterYmax <- ymax(rBrickInput)

##############################################################################################################
# create a function that generates SI noise for each band of the imagery.
addSINoise <- function(SINoise){
  #determine the appropriate SD  for the signal independent noise
  #first determine which band in the input is the defined SNR value
  SIBandPosition <- which(abs(outputWavelengthPositions-highMeasuredSNRPosition)==min(abs(outputWavelengthPositions-
                                                        highMeasuredSNRPosition)))
  rSIband <- subset (rBrickInput, SIBandPosition)
  maxSIband <- cellStats ((rSIband), max,  na.rm=TRUE)
  requiredSImean <- maxSIband/highMeasuredSNR

  # apply the noise to each layer
  for (i in 1:nlayers(rBrickInput)){
    rBrickInput1 <- subset (rBrickInput,i)
    noise = matrix(rnorm(ncell(rBrickInput1), mean=0, sd=1), nrow = nrow(rBrickInput1), ncol = ncol (rBrickInput1))
    noise = raster(noise) #convert this matrix to a raster
    extent(noise)= rBrickInput # set the extents of the newly created raster to that of the input data
    noise2 <- (noise)^2
    noise2 <- sqrt(noise2)
    meanNoise <- (cellStats ((noise2), mean,  na.rm=TRUE))
    scalingFactor <- (meanNoise)/(requiredSImean)
    rSDNoiseSubset <- scalingFactor * noise
    rBrickNoise <- addLayer(rBrickNoise, rSDNoiseSubset)   # stack each of the noisy bands
    #rBrickNoise <- addLayer(rBrickNoise, noise)   # stack each of the noisy bands
  }

  rBrickNoise <- dropLayer(rBrickNoise, 1)   # drop the first layer as it was temporary
  return (rBrickNoise)
}

# create a function that generates SD noise for each band of the imagery.
addSDNoise <- function(SDNoise){

  # determine the intensity of noise to apply based on each layer
  for (i in 1:nlayers(rBrickInput)){
    rBrickInputi <- subset (rBrickInput,i)
    #rBrickInputi [is.nan(rBrickInputi)] = 0
    #rBrickInputi [is.infinite(rBrickInputi)] = 0
    #rBrickInputi [is.na(rBrickInputi)] = 0
```

```
  rBrickInputisqrt <- sqrt (rBrickInputi)
  matrixBrickSpectral1 = as.matrix(rBrickInputisqrt, nrow = (matrixNrow), ncol = (matrixNcol) ) #convert the input
  # raster to a matrix
  tempNumberic1 <- as.numeric(matrixBrickSpectral1) #now convert it to a vector
  #change any value that is negeative to a 0 value. Values are used to set the standard devation in the random noise.
  # Since negetative values cannot be used as a SD then it must be changed to 0
  tempNumberic1 [tempNumberic1 < 0] <- 0
  #change any value that is NA to a 0 value. Values are used to set the standard devation in the random noise. Since
  # NA values cannot be used as a SD then it must be changed to 0
  tempNumberic1[is.na(tempNumberic1[])] <- 0
  tempNumberic2 <- (tempNumberic1[1]) #now convert it to a vector with just the first number

  for (p in 1:length(tempNumberic1)){ #for each pixel generate a random value based on gaussian distribution with the
    # SD being equal to the sqrt of the pixel values.
    SD <- rnorm(1, mean=0, sd=tempNumberic1[p])
    tempNumberic2 <- append(tempNumberic2, SD)
  }
  tempNumberic2  <- tail(tempNumberic2, -1) #drop the first value in the vector. It was temporary.
  matrixBrickSpectral2 <- matrix(tempNumberic2, nrow = (matrixNrow), ncol = (matrixNcol)) #convert the vector back to
  # a matrix
  #print (max(matrixBrickSpectral2))
  rBrickNoisei <-raster (matrixBrickSpectral2, xmn=rasterXmin, xmx=rasterXmax, ymn=rasterYmin, ymx=rasterYmax)#convert
  # the matrix back to a raster
  rBrickNoise <- addLayer(rBrickNoise, rBrickNoisei)   # stack each of the noisy bands
 }
 rBrickNoise <- dropLayer(rBrickNoise, 1)   # drop the first layer as it was temporary
 return (rBrickNoise)
}
##############################################################################################
# Run Signal Dependent Noise Function
noisetype <- 'signalDependent'
rSDNoise <- addSDNoise(x)
##############################################################################################
# Run Signal Independent Noise Function
noisetype = 'signalIndependent'
rSINoise <- addSINoise(x)
##############################################################################################
#Ensure that NA values are assigned 0
rBrickInput [rBrickInput==-0] <-NA
rSDNoise[is.na(rSDNoise[])] <- 0
rSINoise[is.na(rSINoise[])] <- 0
rSINoise <- mask (rSINoise,rBrickInput) # mask additive noise in areas where there are NA values in the input imagery.
rSDNoise <- mask (rSDNoise,rBrickInput) # mask additive noise in areas where there are NA values in the input imagery.

# Create raster bricks that will house the noise, output, and signal dependent noise
rOutput1 <- subset(rBrickInput,1)
rBrickNoise <- subset(rBrickInput,1)
# create a false first band for the SD noise raster brick
rSDNoise1 <- subset(rSDNoise,1)
rSDNoise <- stack(rSDNoise1, rSDNoise)
# create a false first band for the SD noise raster brick
rSINoise1 <- subset(rSINoise,1)
rSINoise <- stack(rSINoise1, rSINoise)
# create a false first band for the input raster brick
rBrickInput1 <- subset(rBrickInput,1)
rBrickInput <- stack(rBrickInput1, rBrickInput)
element2Add <- 418 # a false wavelength element to be added tempoarily
outputWavelengthPositionsTemp <- append (element2Add,outputWavelengthPositions)

##############################################################################################
#calculate the scaling factor needed to change the signal Dependent noise to the desired SNR when combining noise with
# imagery.
#The following lines calculate how much the signal Dependent noise has to be scaled based on the existing SNR from the
# signal depenent noise
##############################################################################################
#linearly interpolate the desired SNR value for each band
for (i in (outputWavelengthPositionsTemp)){
 # apply the linear interpolation equation
 # the variables for the linear interpolation are explained as follows:
 # x = desired wavelength
 # x1 = wavelength position of the known SNR for the VNIR
 # x2 = wavelength position of the known SNR for the SWIR
 # y = desired values at the required wavelength
 # y1 = SNR of the VNIR
 # y2 = SNR of the SWIR
 x <- i
 j <- which (outputWavelengthPositionsTemp==i)
 #if (j > (nlayers (rBrickInput))) break # stop the looping if there are more wavelength records
 x1 <- desiredSNRVNIRWavelength
 x2 <- desiredSNRSWIRWavelength
```

```
y1 <- desiredSNRVNIR
y2 <- desiredSNRSWIR
x3 = (x - x1)
y3 = (y2 - y1)
x4 = (x2 + x1)
xy1 = x3 * y3
xy2 = xy1 / x4
y = y1 + xy2
y <- y1 + ((x - x1)*((y2 - y1)/(x2 - x1)))
desiredSNR <- y

rInputSubset <- subset(rBrickInput,j)
k <- j

#Subset all data so that the SNR is calculated based on the reflectance value at the SNR parameter
rSINoiseSubset <- subset(rSINoise,k)
rSDNoiseSubset <- subset(rSDNoise,k)
rInputSubsetSelect <- rInputSubset
rSINoiseSubsetSelect <- rSINoiseSubset
rSDNoiseSubsetSelect <- rSDNoiseSubset

#scale noise according to the need to fulfil the desired SNR
rInputSubsetSelect2 <- (rInputSubsetSelect)^2
rInputSubsetSelect2 <- sqrt(rInputSubsetSelect2)
meanrBrickSubset <- (cellStats ((rInputSubsetSelect2), mean,  na.rm=TRUE))

rSINoiseSubsetSelect2 <- (rSINoiseSubsetSelect)^2
rSINoiseSubsetSelect2 <- sqrt(rSINoiseSubsetSelect2)
meanrSINoiseSubset <- (cellStats ((rSINoiseSubsetSelect2), mean,  na.rm=TRUE))
rSDNoiseSubsetSelect2 <- (rSDNoiseSubsetSelect)^2
rSDNoiseSubsetSelect2 <- sqrt(rSDNoiseSubsetSelect2)
meanrSDNoiseSubset <- (cellStats ((rSDNoiseSubsetSelect2), mean,  na.rm=TRUE))
scalingFactor <- ((meanrBrickSubset / (desiredSNR))- meanrSINoiseSubset)/meanrSDNoiseSubset
print (desiredSNR)
cat ("scaling factor :", scalingFactor)
rSDNoiseSubset <- scalingFactor * rSDNoiseSubset

#calculate the SNR after appliying the scaling factor
rSINoiseSubsetSelect <- rSINoiseSubset
rSDNoiseSubsetSelect <- rSDNoiseSubset
rSINoiseSubsetSelect [is.na(rInputSubsetSelect)] = NA
rSDNoiseSubsetSelect [is.na(rInputSubsetSelect)] = NA
rSDNoiseSubset2 <- ((rSDNoiseSubset)^2)
rSDNoiseSubset2 <- sqrt(rSDNoiseSubset2)
meanSDNoiseSubset <- cellStats ((rSDNoiseSubset2), mean,  na.rm=TRUE)

#add the two noise sources together
rNoiseSubset <- rSDNoiseSubset + rSINoiseSubset

#calculate the SNR after applying the scaling factor
rNoiseSubsetSelect <- rNoiseSubset
rNoiseSubsetSelect [is.na(rInputSubsetSelect)] = NA
rNoiseSubsetSelect2 <- ((rNoiseSubsetSelect)^2)
rNoiseSubsetSelect2 <- sqrt(rNoiseSubsetSelect2)
meanNoiseSubset <- cellStats ((rNoiseSubsetSelect2), mean,  na.rm=TRUE)

#reapply scaling factor if necessary
scalingFactor<- (meanrBrickSubset/desiredSNR)/meanNoiseSubset
print (scalingFactor)
rNoiseSubset <- scalingFactor * rNoiseSubset

#Recalculate the SNR after applying the scaling factor
rNoiseSubsetSelect <- rNoiseSubset
rNoiseSubsetSelect [is.na(rInputSubsetSelect)] = NA
rNoiseSubsetSelect2 <- ((rNoiseSubsetSelect)^2)
rNoiseSubsetSelect2 <- sqrt(rNoiseSubsetSelect2)
meanNoiseSubset <- cellStats ((rNoiseSubsetSelect2), mean,  na.rm=TRUE)

#Calculate the current SNR
SNR <- meanrBrickSubset/meanNoiseSubset
SNR <- round(SNR)
y <- round(y)
#cat ("\nProcessing wavelength :", x, "(Band :", paste (j),")")
#cat ("\nfor which there is an interpolated SNR of :", y, "and processed as:", paste (SNR))
rOutputSubset <- rInputSubset + rNoiseSubset

rOutput1 <- addLayer(rOutput1,rOutputSubset)
rBrickNoise <- addLayer(rBrickNoise,rNoiseSubset)
rSINoise1 <- addLayer(rSINoise1, rSINoiseSubset)
rSDNoise1 <- addLayer(rSDNoise1, rSDNoiseSubset)
```

```
}

# drop the first band from each rBrick as they were created earlier as a temporary band
rBrickOutput <- dropLayer(rOutput1,1)
rBrickNoise <- dropLayer(rBrickNoise,1)
rSDNoise <- dropLayer(rSDNoise1,1)
rSINoise <- dropLayer(rSINoise1,1)
rBrickInput <- dropLayer(rBrickInput, 1)
rBrickOutput [rBrickInput==0] <-NA
rBrickNoise <- mask (rBrickNoise,rBrickOutput) # mask additive noise in areas where there are NA values in the input
# imagery.
rBrickOutput <- mask (rBrickOutput,rBrickOutput) # mask additive noise in areas where there are NA values in the input
# imagery.
rBrickOutput [rBrickOutput==0] <-NA

#calculate the SNR for both the signal Independent and signal Dependent (total additive) noise
desiredSNRVNIRWavelengthPosition <- which (outputWavelengthPositions==(desiredSNRVNIRWavelength))
rBrickSpectralVNIR <- subset(rBrickInput, (desiredSNRVNIRWavelengthPosition))
noiseVNIR <- subset(rBrickNoise, (desiredSNRVNIRWavelengthPosition))
noiseVNIR [is.na(rBrickSpectralVNIR)] <-NA
rBrickSpectralVNIR2 <- (rBrickSpectralVNIR)^2
rBrickSpectralVNIR2 <- sqrt(rBrickSpectralVNIR2)
noiseVNIR2 <- (noiseVNIR)^2
noiseVNIR2 <- sqrt(noiseVNIR2)
SNRVNIR <- cellStats ((rBrickSpectralVNIR2), mean,  na.rm=TRUE)/(cellStats ((noiseVNIR2), mean,  na.rm=TRUE))
SNRVNIR3 <- cellStats ((rBrickSpectralVNIR2), mean,  na.rm=TRUE)/(cellStats ((rBrickSpectralVNIR2), sd,  na.rm=TRUE))

desiredSNRSWIRWavelengthAdd <- desiredSNRSWIRWavelength + 0.001
desiredSNRSWIRWavelengthPosition <- which (outputWavelengthPositions==(desiredSNRSWIRWavelengthAdd))
rBrickSpectralSWIR <- subset(rBrickInput, (desiredSNRSWIRWavelengthPosition))
noiseSWIR <- subset(rBrickNoise, (desiredSNRSWIRWavelengthPosition))
noiseSWIR [is.na(rBrickSpectralSWIR)] <-NA
rBrickSpectralSWIR2 <- (rBrickSpectralSWIR)^2
rBrickSpectralSWIR2 <- sqrt(rBrickSpectralSWIR2)
noiseSWIR2 <- (noiseSWIR)^2
noiseSWIR2 <- sqrt(noiseSWIR2)
SNRSWIR <- cellStats ((rBrickSpectralSWIR2), mean,  na.rm=TRUE)/(cellStats ((noiseSWIR2), mean,  na.rm=TRUE))

# make a non-overlapping raster brick to export
nVNIRBands <- floor((VNIRmaximumWavelength - VNIRminimumWavelength)/desiredSpectralResVNIR) #number of VNIR bands
# (rounding up)
SWIRmaximumWavelength <- SWIRmaximumWavelength-SWIRBuffer
SWIRminimumWavelength <- SWIRminimumWavelength+SWIRBuffer+1
nSWIRBands <- floor((SWIRmaximumWavelength - SWIRminimumWavelength)/desiredSpectralResSWIR) #number of SWIR bands
# (rounding up)
nVNIRBands <- nVNIRBands + 1
nSWIRBands <- nSWIRBands + 1
wavelengthOverlapOut <- append(wavelengthVNIROut, wavelengthSWIROut)

# figure out which band starts the nonoverlapping section of the SWIR
wavelengthSWIRBands <- outputWavelengthPositions [(nVNIRBands+1):length(outputWavelengthPositions)]
SWIRStartPosition <- nVNIRBands + which.min(abs(VNIRmaximumWavelength - wavelengthSWIRBands))
SWIRStartPosition <- SWIRStartPosition + 1
rBrickOutputNonOverlapVNIR <- subset (rBrickOutput, 1:nVNIRBands)
rBrickOutputNonOverlapSWIR <- subset (rBrickOutput, SWIRStartPosition:nlayers (rBrickOutput))
rBrickOutputNonOverlap <- stack(rBrickOutputNonOverlapVNIR,rBrickOutputNonOverlapSWIR)
wavelengthSWIROut <- wavelengthSWIROut - 0.001 #remove the previously added 0.001 which was done in the spectral
# convolution to create unique numbers
wavelengthOverlapOut <- append(wavelengthVNIROut, wavelengthSWIROut)

# Remove any extra records in the FWHMNonOverlapOut vector if they exist.
if (length (FWHMNonOverlapOut)>nlayers (rBrickOutputNonOverlap)){
  repeat {
    if (length (FWHMNonOverlapOut) == nlayers (rBrickOutputNonOverlap)){
      break
    }
    FWHMNonOverlapOut <- head (FWHMNonOverlapOut,(length(FWHMNonOverlapOut)-1))
  }
}

#Remove any extra bands if they exist beyond the desired maximum wavelength and also remove from the wavelength vector
# to be used for populating metadata
wavelengthNonOverlapOut <- append(wavelengthVNIROut, (tail(wavelengthSWIROut,nlayers(rBrickOutputNonOverlapSWIR))))
wavelengthNonOverlapOut <- round(wavelengthNonOverlapOut/0.5)*0.5 # round the wavelength numbers back to their original
# half value
if (tail (wavelengthNonOverlapOut,1)>SWIRmaximumWavelength){
  repeat {
    if (tail (wavelengthNonOverlapOut,1)==SWIRmaximumWavelength){
      break
    }
```

```
    wavelengthNonOverlapOut <- head (wavelengthNonOverlapOut,(length(wavelengthNonOverlapOut)-1))
    wavelengthOverlapOut <- head (wavelengthOverlapOut,(length(wavelengthOverlapOut)-1))
    rBrickOutputNonOverlap <- dropLayer (rBrickOutputNonOverlap,nlayers (rBrickOutputNonOverlap))
    rBrickOutput <- dropLayer (rBrickOutput,nlayers (rBrickOutput))
    rSINoise <- dropLayer (rSINoise,nlayers (rSINoise))
    rSDNoise <- dropLayer (rSDNoise,nlayers (rSDNoise))
    rBrickNoise <- dropLayer (rBrickNoise,nlayers (rBrickNoise))
  }
}

# Create an external .CSV file with the wavelength information of the non-overlapping datacube for attributing the
#metadata within
# ENVI or similar.
write.table(wavelengthNonOverlapOut, file = "NonOverlap_wavelength_band_Info.csv", sep = ",", col.names = FALSE,
          qmethod = "double")
# Create an external .CSV file with the FWHM information of the non-overlapping datacube for attributing the metadata
#within ENVI or similar.
FWHMVNIROut <- rep(desiredSpectralResVNIR, nlayers (rBrickOutputNonOverlapVNIR))
FWHMSWIROut <- rep(desiredSpectralResSWIR, nlayers (rBrickOutputNonOverlapSWIR))
FWHMNonOverlapOut <- append(FWHMVNIROut, FWHMSWIROut)
write.table(FWHMNonOverlapOut, file = "NonOverlap__FWHM_band_Info.csv", sep = ",", col.names = FALSE, qmethod =
              "double")

# Remove any extra records in the FWHMOverlapOut vector if they exist.
if (length (FWHMOverlapOut)>nlayers (rBrickOutput)){
  repeat {
    if (length (FWHMOverlapOut) == nlayers (rBrickOutput)){
      break
    }
    FWHMOverlapOut <- head (FWHMOverlapOut,(length(FWHMOverlapOut)-1))
  }
}

# Create an external .CSV file with the wavelength information of the overlapping datacube for attributing the metadata
# within ENVI or similar.
write.table(wavelengthOverlapOut, file = "Overlap_wavelength_band_Info.csv", sep = ",", col.names = NA, qmethod =
              "double")

# Create an external .CSV file with the FWHM information of the non-overlapping datacube for attributing the metadata
# within ENVI or similar.
FWHMVNIROut <- rep(desiredSpectralResVNIR, nlayers (spectralConvRasterVNIR))
FWHMSWIROut <- rep(desiredSpectralResSWIR, nlayers (spectralConvRasterSWIR))
FWHMNonOverlapOut <- append(FWHMVNIROut, FWHMSWIROut)
write.table(FWHMOverlapOut, file = "Overlap__FWHM_band_Info.csv", sep = ",", col.names = FALSE, qmethod = "double")

#Output the files
outputfilename <- "SimulatedDatacubeNoOverlappingBands.tif" # * change the name as needed
print ("Creating output file...")
output <- writeRaster(rBrickOutputNonOverlap, filename=(outputfilename), overwrite=TRUE)
outputfilename <- "SimulatedDatacube.tif" # * change the name as needed
print ("Creating output file...")
output <- writeRaster(rBrickOutput, filename=(outputfilename), overwrite=TRUE)
outputfilename <- "SINoise.tif" # * change the name as needed
print ("Creating output file...")
output <- writeRaster(rSINoise, filename=(outputfilename), overwrite=TRUE)
outputfilename <- "SDNoise.tif" # * change the name as needed
print ("Creating output file...")
output <- writeRaster(rSDNoise, filename=(outputfilename), overwrite=TRUE)
outputfilename <- "TotalNoise.tif" # * change the name as needed
print ("Creating output file...")
output <- writeRaster(rBrickNoise, filename=(outputfilename), overwrite=TRUE)

cat ("The SNR for the total noise at",desiredSNRVNIRWavelength ,"nm VNIR is ", paste (SNRVNIR))
cat ("The SNR for the total noise at",desiredSNRSWIRWavelength ,"nm SWIR is ", paste (SNRSWIR))

print ("The user may wish to add the metadata information to the output datacubes. In ENVI classic this is achieved by
      # editing the ENVI header file and importing the information stored in the appropriate corresponding .CSV files
      # for the wavelength and FWHM information created as an output of this script.")
print ("  ")
print ("Users of this script may also notice when viewing the output datacubes in ENVI that the no data areas influence
      the statistics of the data such that it causes the data to display as a white object with no contrast between
      values. This is a limitation of the outputs created by R. This can be overcome in ENVI by building and applying a no
      data mask to the output data.")

###############################################################################################
# #stop the stopwatch to see how long the script took to run
proc.time() - starttime
print (proc.time() - starttime)
# end of script
###############################################################################################
```

Appendix C: CRAN R Script for Thresholding Matched Filtered Score Images and Conversion to a Multiclass Map

```
#############################################################################################
# Binary classification based on Max Probability.R
# Script details:
# This script produces a binary hard classification image using class probabilities, MF scores, etc. images.
#
# Prior to using this script all packages and libraries have to be loaded using the script:
# "1. Load Scripts.r".
#
# The user can search the script for the special character '*' to find lines that are
# modifiable variables.
#
# Written by Roger MacLeod - University of Victoria and the Geological Survey of Canada -
# Natural Resources Canada
#
# Script completed December, 2016, 2017
#############################################################################################
#rm(list=ls()) #remove all unnecesary R objects currently loaded
#############################################################
# Do you want to threshold the values based on the mean value or the SD?
thresholdMethod <- 'SD' # enter either 'SD' or 'Mean' *
# Do you want to threshold the values based on the stats of each band rather than all bands?
thresholdClass <- 'Y' #enter 'Y' or 'N' *
#############################################################
starttime <- proc.time()
setwd("U:/Thesis/Data/ClassifcationResults/MF/HiRes") # *

rBrickInput <- brick("U:/Thesis/Data/MFResults/Ideal5/HiRes/HiResMF.dat", package="raster") # modify value as necessary *

#Specify the input masking image
rMask <- raster("U:/Thesis/Data/Masks/HiResTotalMask.dat", package="raster") # modify value as necessary * Define the
# Masking layer
#############################################################
# Define the Size of the low pass filter kernel to apply ot the input images. A value of 0 means no filtering. *
rnrow <- 0
#############################################################
# Optional Low Pass Filtering to the MF before classifying image. Uncomment this section if user does not wish to use
# Uncomment one of the following lines starting with 'r <- focal' as required for the desired size of kernel.
if (rnrow > 0){ #only apply a low pass filter if the user entered a filter size above
  #print (rnrow)
  tempR <- subset (rBrickInput,1)#create a temporary raster with matching extents and resolution to that of the input
  # data that will hold the outputs
  for (i in 1:nlayers(rBrickInput)) {
    r <- subset (rBrickInput,i) #subset each probability map
    rncol <- rnrow
    rmatrix <- rnrow ^2
    rmatrix <- matrix(1/rmatrix,rnrow,rncol)
    #print (rmatrix)
    r <- focal (r, w=rmatrix, na.rm=T)
    tempR <- addLayer(tempR, r) #stack each of the filtered MF score maps as they are processed
  }
  rBrickInput <-dropLayer(tempR,1)
}

#############################################################
thresholdValue <- 0.5 #set the multiplier of the standard deviation or mean of the images to determine the thresholded
# value for masking the classified map.
thresholdValueUnchanged <- thresholdValue
maxValueRaster <-calc(rBrickInput,fun=max) # create a single raster map with the maximum values of all bands
whichR <- which.max (rBrickInput) ### number of the layer with maximum value
tempR <- subset (rBrickInput,1)#create a temporary raster with matching extents and resolution to that of the input data
# that will hold the outputs
if (thresholdMethod == 'Mean'){
  thresholdValueAllBands <- (thresholdValue)*(mean(cellStats (rBrickInput, 'mean')))#this line determines the threshold
  # value based on the average of the input data
  print ("threshold values are:")
  print (thresholdValueAllBands)
} else {
  thresholdValueAllBands <- (thresholdValue)*(mean(cellStats (rBrickInput, 'sd')))#this line determines the threshold
  # value based on the average of the input data
}

for (i in 1:nlayers(rBrickInput)) { #create a loop that evaluates each probability map and creates a rasterbrick that
  # is thresholded
  r <- subset (rBrickInput,i) #subset each probability map
  #print ((cellStats (r, 'mean')))
  #print ((cellStats (r, 'sd')))
  if (thresholdClass == 'Y'){
    if (thresholdMethod == 'Mean'){
      thresholdValue <- (thresholdValueUnchanged)*(cellStats (r, 'mean'))
      } else {
```

```
    thresholdValue <- (thresholdValueUnchanged)*(cellStats (r, 'sd'))
   }
  } else {
   thresholdValue <- thresholdValueAllBands
  }

  r[r < (thresholdValueUnchanged)] <- NA #check that the probabilty value is higher than the user defined threshold
  # value
  r[maxValueRaster != r] <- NA #check if the probability value is the highest among all maps and change pixel values
  # to NA if they aren't. In other words the probability maps are masked
  whichR[(is.na(r))] <- NA # change the classification map so that the threholded pixels are masked
  tempR <- addLayer(tempR, r) #stack each of the thresholded and masked probabilty maps
}
thresholdedProbability <-dropLayer(tempR,1) # drop the first raster as it was created temporarily earlier.
gc()

#Create the classified maps from the thresholded probability maps
r2 <- subset (thresholdedProbability,1)
r2[(is.na(r2))] <- 0
r2 [r2] <-0

for (i in 1:nlayers(thresholdedProbability)) { #create a loop that evaluates each probability map
  r <- subset (thresholdedProbability,i)
  r[(is.na(r))] <- 0
  r2 <- r2 +  r
}
gc()
finalProbabiltyMap <- r2
finalClassifiedMap <- subset (thresholdedProbability,1)
gc()

finalClassifiedMap[] <- 0
finalClassifiedMap[(is.na(finalClassifiedMap))] <- 7 #* change as needed to one above the number of classes
# you now need to write the script to evaluate which band the threshold value came from
for (i in 1:nlayers(rBrickInput)) { #create a loop that evaluates the final probability map for which class the highest
  # value came from
  r <- subset (thresholdedProbability,i)
  finalClassifiedMap[r == finalProbabiltyMap] <- i
}
gc()

#Msak all outputs based on teh Total Mask derived through water and vegeation masking
finalClassifiedMap [rMask == 0] <-0 #
finalProbabiltyMap [rMask == 0] <-0 #

# #Plot the classified Map
# #First create rasters for each class
finalClassifiedMap.1 <- calc(finalClassifiedMap, fun=function(x){ x[x != 1] <- NA; return(x)} )
finalClassifiedMap.2 <- calc(finalClassifiedMap, fun=function(x){ x[x != 2] <- NA; return(x)} )
finalClassifiedMap.3 <- calc(finalClassifiedMap, fun=function(x){ x[x != 3] <- NA; return(x)} )
finalClassifiedMap.4 <- calc(finalClassifiedMap, fun=function(x){ x[x != 4] <- NA; return(x)} )
finalClassifiedMap.5 <- calc(finalClassifiedMap, fun=function(x){ x[x != 5] <- NA; return(x)} )
finalClassifiedMap.6 <- calc(finalClassifiedMap, fun=function(x){ x[x != 6] <- NA; return(x)} )

#Now plot the rasters in one plot
#dev.new()
plot(finalClassifiedMap.1, col="red", legend =FALSE)
par(new=TRUE)
plot(finalClassifiedMap.2, col="green", legend =FALSE)
par(new=TRUE)
plot(finalClassifiedMap.3, col="blue", legend =FALSE)
par(new=TRUE)
plot(finalClassifiedMap.4, col="forestgreen", legend =FALSE)
par(new=TRUE)
plot(finalClassifiedMap.5, col="cyan", legend =FALSE)
par(new=TRUE)
plot(finalClassifiedMap.6, col="magenta", legend =FALSE)


#output the classifed images
#outputfilename1 <- "HiResMFProbability3.tif" # *
writeRaster(finalProbabiltyMap, filename=(outputfilename1), overwrite=TRUE)
gc()

#outputfilename3 <- "HiResMFClassifiedMap3.tif" # *
writeRaster(finalClassifiedMap, filename=(outputfilename3), overwrite=TRUE)
gc()

#create a mask image for the no data areas
mask_1 <- finalClassifiedMap
```

```
mask_1 [mask_1 > 0] <-1 #
mask_1 [rMask == 0] <-0 #
gc()

#mask_1 [(is.na(mask_1))] <- 1

#plot (mask_1)
#outputfilename4 <- "HiResMFClassifiedMapMask3.tif" # *
writeRaster(mask_1, filename=(outputfilename4), overwrite=TRUE)
gc()

#############################################################################################
#stop the stopwatch to see how long the script took to run
proc.time() - starttime
print (proc.time() - starttime)
# end of script
#############################################################################################
```

Appendix D: Sherlock and Carpenter's Map (2003): Bedrock Geology of the Wolverine-Doris Corridor, Hope Bay Volcanic Belt, Nunavut (GSC Open File 1553)

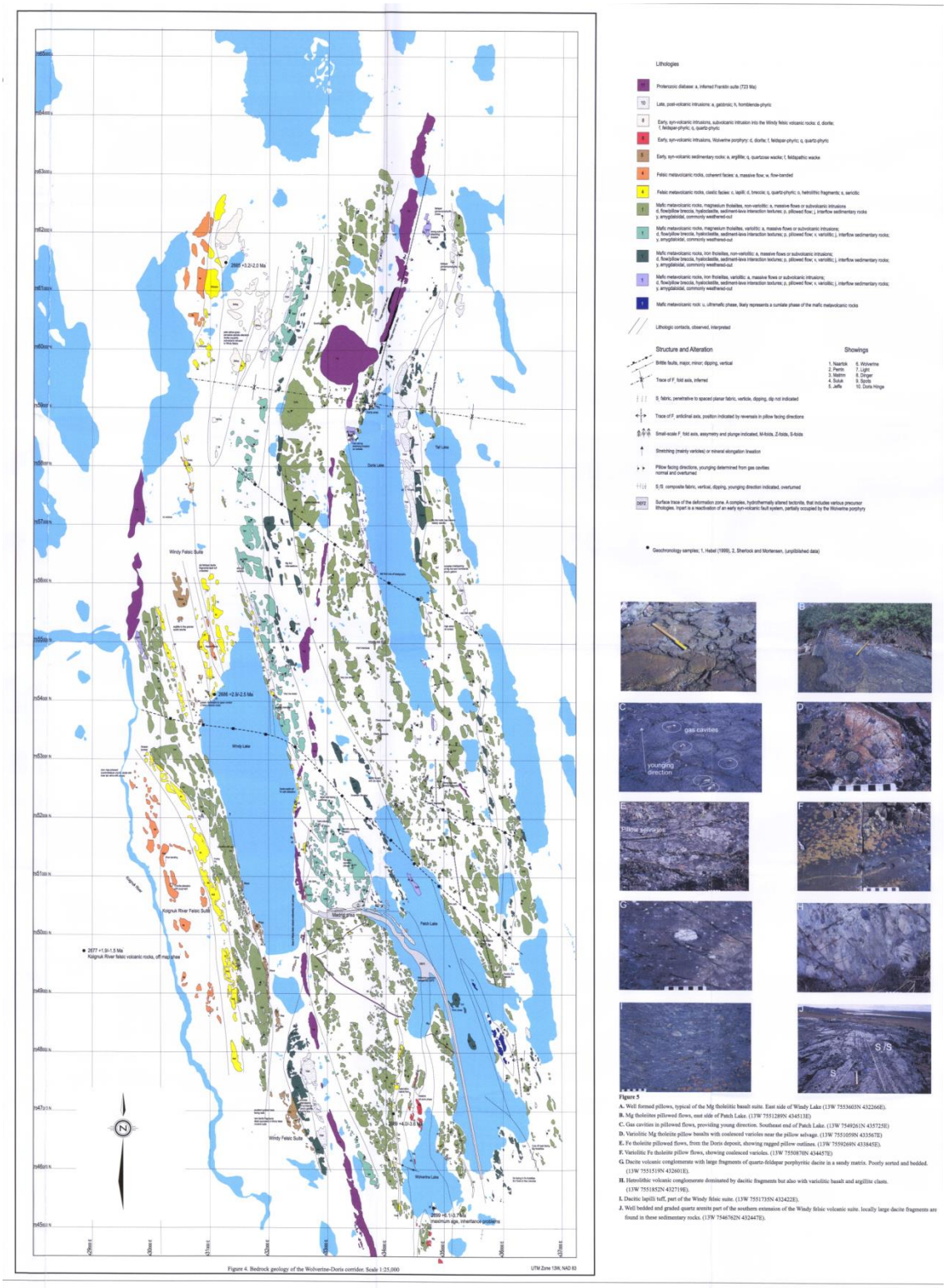

Figure 4. Bedrock geology of the Wolverine-Doris corridor. Scale 1:25,000 UTM Zone 13W, NAD 83

Figure 5

A. Well formed pillows, typical of the Mg tholeiitic basalt suite. East side of Windy Lake (13W 7553603N 432266E).

B. Mg tholeiites pillowed flows, east side of Patch Lake. (13W 7551289N 434513E)

C. Gas cavities in pillowed flows, providing young direction. Southeast end of Patch Lake. (13W 7549261N 435725E)

D. Variolitic Mg tholeiitic pillow breatch with coalesced varioles near the pillow selvage. (13W 7551059N 433567E)

E. Fe tholeiitic pillowed flows, front the Doris deposit, showing ragged pillow outlines. (13W 7550269N 433845E).

F. Variolitic Fe tholeiitic pillow flows, showing coalesced varioles. (13W 7550870N 434457E)

G. Dacite volcanic conglomerate with large fragments of quartz-feldspar porphyritic dacite in a sandy matrix. Poorly sorted and bedded. (13W 7551519N 432601E).

H. Hemolithic volcanic conglomerate dominated by dacitic fragments but also with variolitic basalt and argillite clasts. (13W 7551852N 432719E).

I. Dacitic lapilli tuff, part of the Windy felsic suite. (13W 7551735N 432422E).

J. Well bedded and graded quartz arenite part of the southern extension of the Windy felsic volcanic suite. locally large dacite fragments are found in these sedimentary rocks. (13W 7546762N 432447E).