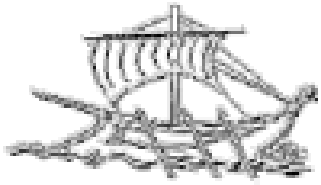


TEI of PIRAEUS
School of Technological Appliance
DEPARTMENT OF ELECTRONICS AND COMPUTER SYSTEMS ENGINEERING

PG COURSE "APPLIED INFORMATION SYSTEMS"

Classification Using Hyper-spectral Imagery

by

Ilias Dimopoulos

Matriculation No. ais-0105

A thesis submitted in partial fulfilment of the requirements for the degree of

Masters in Applied Information Systems

Supervisor: Pr. Anastasia Veloni

ATHENS
DECEMBER 2017

Postgraduate final thesis

Classification Using Hyper-Spectral Imagery

Ilias Dimopoulos
Matriculation No. ais-0105

Supervisor:

Pr. Anastasia Veloni

Examination Committee:

Ioannis Ellinas, Professor

DECLARATION OF DOCUMENTARY AUTHOR

I, the undersigned .....,of ....., with registration number .....student of the Department of Computer Systems Engineering of the T.E.I of Piraeus before I undertake my Masters thesis, I declare that I have been informed of the following:
"The Dissertation is a product of intellectual property of both the author and the Foundation and should have a unique character and original content.
It is strictly forbidden for any piece of text to appear as it is or translated from another published source. Every such act is a product of plagiarism and raises a matter of moral order for the copyright of the other writer. The author is the sole author of the document, who also bears the responsibility for the consequences, criminal and other, of this act.
In addition to any criminal responsibility of the author in the event that the Foundation has awarded him a degree, this is revoked by a decision of the Assembly of the Department. The Assembly of the Department, with a new decision, at the request of the interested party, re-assigns it to the elaboration of the Dissertation with another subject and different supervisor. The elaboration of this Thesis must be completed within at least one six months from the date of its award. For the rest, the provisions of Rule 18, par. 5 of the Rules of Procedure shall apply."

Acknowledgments

The author wishes to thank Pr. Anastasia Veloni (Professor at TEI of Piraeus) for her constant and effective help with programming and development issues during the stages of the project development. Also a very special thanks to my parents for all the encouragement and support they provided me throughout the whole period showing me the way to succeed. Generally, thanks to all people, and especially my friends, who helped me and kept me motivated to continue with my dissertation till the end.

Synopsis

The subject of this dissertation is the study of concepts and principles of Remote Sensing focusing on the analysis and processing of Hyperspectral Images. Hyperspectral imaging as a tool for monitoring, detection and classification of objects and areas on Earth can be used in a wide range of applications. Especially in the case of monitoring environmental and climatic changes, hyperspectral imaging can provide plenty of information in order to create an environmental database develop models and prevent possible natural disasters.

Based on this framework and using Matlab software tool a processing system was proposed and developed. The procedure of processing raw hyperspectral data is described and a classification system is developed. This system uses classification algorithms in order to distinguish areas with vegetation, water resources, dry soil or man-made constructions using hyperspectral images from a remote sensor. Four supervised classification algorithms were developed and evaluated: Euclidean Minimum Distance, Mahalanobis Minimum Distance, Bayesian Classification and Maximum Likelihood approximation. The algorithms were applied on a satellite hyperspectral image of an area. Results have shown that the fastest algorithm is the Euclidean distance giving relatively accurate results. Weaknesses in the performance of the Mahalanobis distance algorithm were reveled in case of insufficient training sets. Bayesian Classification algorithm performed exceptionally well. However the best performance as far as accuracy is concerned were provided by the proposed approximation of the Maximum Likelihood Distance Algorithm, which also reaches satisfactory performance without the need of a long iterative process.

During the study the spectral signatures for the defined classes were estimated. Comparison with existing studies has shown that the estimates are representative of the area under investigation confirming the integrity of the proposed procedure.

The dissertation is partitioned in two sections. Chapters 2, 3, 4 and 5 provide the necessary background and state of the art in the subjects of remote sensing and Hyperspectral Imaging, while chapters 6, 7 and 8 present the classification algorithms, the implemented processing system, the results and conclusions. The Matlab code with necessary clarifications and explanations is also provided in the Appendix.

Scientific area: a satellite hyperspectral image of an area

Keywords: Hyperspectral imaging, Matlab, Euclidean Minimum Distance, Mahalanobis Minimum Distance, Bayesian Classification, Maximum Likelihood approximation, spectral signatures

Σύνοψη

Το αντικείμενο της παρούσας διατριβής είναι η μελέτη των εννοιών και των αρχών της Τηλεπισκόπησης με επίκεντρο την ανάλυση και επεξεργασία των Υπερφασματικών Εικόνων. Η υπερφασματική απεικόνιση ως εργαλείο παρακολούθησης, ανίχνευσης και ταξινόμησης αντικειμένων και περιοχών στη Γη μπορεί να χρησιμοποιηθεί σε ένα ευρύ φάσμα εφαρμογών. Ειδικά στην περίπτωση παρακολούθησης περιβαλλοντικών και κλιματικών αλλαγών, η υπερφασματική απεικόνιση μπορεί να παρέχει πολλές πληροφορίες για τη δημιουργία μιας περιβαλλοντικής βάσης δεδομένων για την ανάπτυξη μοντέλων και την πρόληψη πιθανών φυσικών καταστροφών.

Με βάση αυτό το πλαίσιο και με τη χρήση του εργαλείου λογισμικού Matlab προτάθηκε και αναπτύχθηκε ένα σύστημα επεξεργασίας. Περιγράφεται η διαδικασία επεξεργασίας ακατέργαστων υπερφασματικών δεδομένων και αναπτύσσεται ένα σύστημα ταξινόμησης. Αυτό το σύστημα χρησιμοποιεί αλγόριθμους ταξινόμησης για να διακρίνει περιοχές με βλάστηση, υδάτινους πόρους, ξηρό έδαφος ή τεχνητές κατασκευές χρησιμοποιώντας υπερφασματικές εικόνες από έναν απομακρυσμένο αισθητήρα. Καταρτίστηκαν και αξιολογήθηκαν τέσσερις αλγόριθμοι κατηγοριοποίησης: Euclidean Minimum Distance, Mahalanobis Minimum Distance, Bayesian Classification and Maximum Likelihood approximation. Οι αλγόριθμοι εφαρμόστηκαν σε μια δορυφορική υπερφασματική εικόνα μιας περιοχής. Τα αποτελέσματα έχουν δείξει ότι ο γρηγορότερος αλγόριθμος είναι ο Euclidean distance που δίνει σχετικά ακριβή αποτελέσματα. Οι αδυναμίες στην εκτέλεση του αλγορίθμου απόστασης Mahalanobis αποκαλύφθηκαν σε περίπτωση ανεπαρκών ομάδων εκπαίδευσης. Ο αλγόριθμος Bayesian Classification πραγματοποιήθηκε εξαιρετικά καλά. Ωστόσο, η βέλτιστη απόδοση όσον αφορά την ακρίβεια παρέχεται από την προτεινόμενη προσέγγιση του Αλγόριθμου μέγιστης απόστασης (Maximum Likelihood Distance Algorithm), ο οποίος επίσης επιτυγχάνει ικανοποιητικές επιδόσεις χωρίς την ανάγκη μιας μακράς επαναληπτικής διαδικασίας.

Κατά τη διάρκεια της μελέτης εκτιμήθηκαν οι φασματικές υπογραφές (spectral signatures) για τις καθορισμένες κατηγορίες. Από τη σύγκριση με τις υπάρχουσες μελέτες προέκυψε ότι οι εκτιμήσεις είναι αντιπροσωπευτικές της υπό έρευνα περιοχής που επιβεβαιώνει την ακεραιότητα της προτεινόμενης διαδικασίας.

Η διατριβή χωρίζεται σε δύο ενότητες. Τα κεφάλαια 2, 3, 4 και 5 παρέχουν το απαραίτητο υπόβαθρο και την τεχνολογία στα θέματα της τηλεπισκόπησης και της υπερφασματικής απεικόνισης, ενώ στα κεφάλαια 6, 7 και 8 παρουσιάζονται οι αλγόριθμοι ταξινόμησης, το εφαρμοσμένο σύστημα επεξεργασίας, τα αποτελέσματα και τα συμπεράσματα. Ο κώδικας Matlab με τις απαραίτητες διευκρινίσεις και επεξηγήσεις παρέχεται επίσης στο προσάρτημα.

Contents

Table of Figures

List of Tables

1. Introduction

Remote sensing is the procedure of data collection and information acquisition about an object, an area or a phenomenon without making physical contact. It has become a very important tool in many research activities for many scientific fields. However the term remote sensing is usually used to describe the process of monitoring, detecting and classifying objects and areas on Earth (on the surface, in the atmosphere or in the sea)

A Remote Sensing system accomplishes its objective with the use of an array of sensors that measure the reflected or emitted electromagnetic radiation from target areas/objects of Earth. The information recorded is processed and the results of the analysis are used for decision making about the condition, evolution or possible variations of the monitored area/object. Thus Remote Sensing is actually an application of Spectroscopy, the study of the interaction of incident radiation on an object. In most of the cases the radiation source is the Sun and the Earth itself.

Hyper-spectral imaging and the development of hyper-spectral sensors caused the most important breakthrough in remote sensing. Moreover the progress in computer processors allowed the development of fast and efficient algorithms for processing hyper-spectral data. Hyper-spectral imaging can be defined as the acquisition of an object image by analyzing the received radiation in a large frequency range and for a great number of distinct wavelengths. Hyper-spectral Images are a relatively new technology and they are mainly used for detection and identification of minerals, vegetation, resources and features of the surface or the oceanic environment. Depending on the spectral band analyzed by the sensor, the bandwidth resolution, the noise level and other parameters researchers can have a detailed and accurate view of an area of the Earth with the use of data acquired from e.g. a satellite.

In order to be able to monitor areas and phenomena, detect objects and materials it is necessary to develop classification algorithms that will allow the researchers to extract information, track variations and prevent disturbing or catastrophic phenomena. Classic classification algorithms of pattern recognition can be applied to hyper-spectral data: Minimum Distance, Maximum Likelihood, Bayes, Spectral Angle mapper etc.

The benefits of the use of hyper-spectral images in remote sensing are quite obvious. They can be extremely useful in mineralogy and geology and they can be an

excellent tool for mapping environmental changes and phenomena. Climate changes can be observed by mapping vegetation, measuring moisture or the range of barren soils, monitoring the water resources etc. Human intervention in the environment can also be quantified. Therefore Hyper-spectral Images can be used to detect environmental changes and prevent phenomena that can damage the balances of an ecosystem.

The main objective of this dissertation is the study of remote sensing and hyper-spectral images and the development of classification algorithms for monitoring a location from a satellite image. By processing a hyper-spectral image, the task is to identify areas with or without vegetation, human intervention or water resources. The feature extraction from the hyper-spectral data gives the necessary information to develop algorithms that can monitor basic environmental conditions for the specific depicted area. The algorithms studied were implemented using a popular software tool MATLAB and the results of the classification procedures as well as the full list of coding are presented.

In chapter 2, the necessary theoretical background is presented covering basic aspects and definitions about Spectroscopy, Electromagnetic Radiation and Remote Sensing. In chapter 3 descriptions and details about Hyper-spectral Imaging are provided. In chapters 4, 5 and 6 we focus on the processing procedure of hyper-spectral data: Preprocessing, management of hyper-spectral classification processes and Classification Algorithms. The results of the study are used in chapter 7 for the development of MATLAB algorithms for a specific provided hyper-spectral image and results are extracted. Finally in chapter 8 provides concluding remarks within which the outcomes of the overall project will be discussed.

Section A: Background

2. Remote Sensing

A constantly increasing interest in hyper-spectral imaging is observed during the recent years with a plethora of applications in various scientific fields such as medicine, biochemistry, engineering and environmental studies. Until the beginning of the previous decade, only specialists in the field of spectral remote sensing had access in hyper-spectral images and proper software tools to process the received information. Today the analysis of the hyper-spectral images has become one of the most powerful and developing tools in remote sensing.

The term “hyper” in the hyperspectral imaging refers to the vast number of different wavelengths that can be measured and stored in this kind of images. Hyper-spectral data consist of measurements from a large number of spectral zones with relatively small bandwidth each. For example the sensor of the Airborne Visible/Infrared Imaging Spectrometer (AVIRIS [36], [72]) collects simultaneously data from 224 different spectral zones with bandwidth of 0.4 to 2.5 μm and spatial resolution of 20 meters.

Hyper-spectral images provide a great set of data to be used for classification and recognition of objects and materials giving a more accurate perspective for the extraction of specific information than any other type of remote sensing data collection. Classification for a hyperspectral image is the procedure of the creation of an information-based representation of the image content in order to provide the spatial distribution and location of objects or materials with special properties in a studied region.

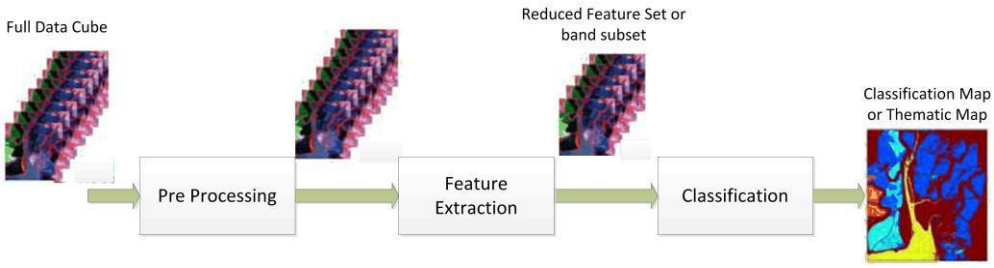


Figure 1: A typical classification system of hyperspectral data

Figure 1 shows the flow chart for a hyperspectral imaging analysis and classification system. In this figure we can distinguish three basic processing steps: a) preprocessing of data, b) feature extraction and finally c) classification [40].

The preprocessing step precedes the main data processing procedures and it includes the isolation of a specific area, noise reduction, enhancements of a specific image view etc. There is a large number of preprocessing procedures that can be applied in the image depending on the application. Some examples for major remote sensing projects are the radiometric calibration, atmospheric correction, enhancement of contrast and image registration.

Feature extraction is an optional step of the process and its objective is to reduce the spectral or spatial dimensions of the image. This is achieved with the selection of an optimum subset of bands in order to avoid estimation problems that can be caused from the high correlation of specific bands or areas. In this stage the hyperspectral image has been transformed in a "feature image". The feature extraction algorithms must be designed so that the needed information of the specific application is kept in order to apply the classification algorithm, while all the unnecessary information is deleted.

Classification of hyperspectral data is the process of identification and registration of all or some image pixels in a specific class of objects or materials based on the statistical properties of the pixel intensities. Classification can be further distinguished in two categories, Supervised and Unsupervised. The classified data must be checked and certified in order to determine the algorithm accuracy and reliability. In the following chapters we will describe with details the total procedure of classification in hyperspectral images and we will perform tests and simulations for some of those algorithms.

2.1 Spectroscopy

In order to fully understand the advantages of hyperspectral imaging it is useful to briefly mention some key issues of spectroscopy and the interaction between matter and electromagnetic radiation.

2.1.1. Electromagnetic Radiation

Electromagnetic radiation is the energy emitted and absorbed by charged particles travelling through space in the form of self-propagating transverse oscillating waves of electric and magnetic field (E and M respectively). The electric and magnetic fields are perpendicular to the wave energy direction and orthogonal to each other. The wave can by described by its wavelength which corresponds to the

physical length of a full wave oscillation and frequency which expresses the number of wave oscillations per second [60].

Electromagnetic radiation presents both wave and particle properties. The dual nature of the radiation, particle and wave, not mutually excluding but complementary, was first discovered from Einstein who expressed radiation as a continuous stream of discrete particles or wave energy "packages" that were later named photons.

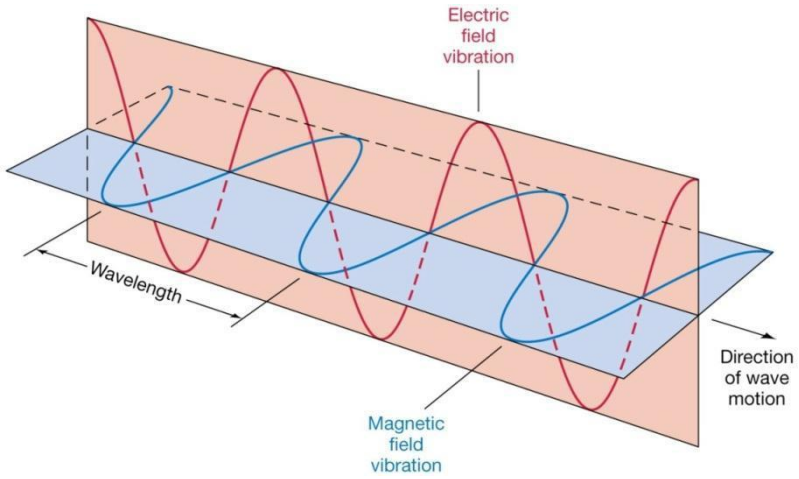


Figure 2: The electromagnetic wave travels through space with the speed of light $c = 2.99792 \times 10^8$. The formula connecting wavelength, frequency and speed of light is: $c = \lambda v$

The energy E of a photon depends on the frequency (or equivalently the wavelength) of the radiation wave:

$$E = hv = {}^{hc}\!/_{\lambda}$$

where is the Planck constant (6.62618x10-34 Js), the speed of light, the frequency and the wavelength. Figure 2 was extracted from [60].

2.2 Electromagnetic Spectrum

The electromagnetic spectrum is the whole range of possible values for the wavelength of the electromagnetic waves and it can extend from extremely large frequencies where $\lambda \to 0$ to extremely small (where the wavelength approaches the universe dimensions). For convenience the electromagnetic spectrum is divided in spectral bands due to the fact that the energy for each band interacts with matter in very different manner.

A spectral band consists of a defined (usually continuous) set of spectral lines, where each spectral line corresponds to a specific wavelength. One simple way

to distinguish the separate spectral bands depending on the wavelength is the following [46]:

Spectral Bands:	Wavelengths:
Gamma Rays	<0.03 nm
X Rays	0.03 – 300 nm
Ultraviolet	0.3 – 0.38 μm
Visible	0.38 – 0.72 μm
Near infrared	0.72 – 1.30 μm
Infrared (short – mid – long)	1.30 – 15 μm
Far infrared	15 – 1000 μm
Microwaves	0.3 – 30 cm
Radiowaves	>30 cm

Table 1: For the sun the spectrum extends from gamma rays (small wavelength – high energy) to radiowaves (large wavelength and low energy)

The main divisions of the electromagnetic spectrum are in essence arbitrarily defined since among the spectral bands many subdivisions can be defined and the transition from one spectral band to the other is gradual rather than steep.

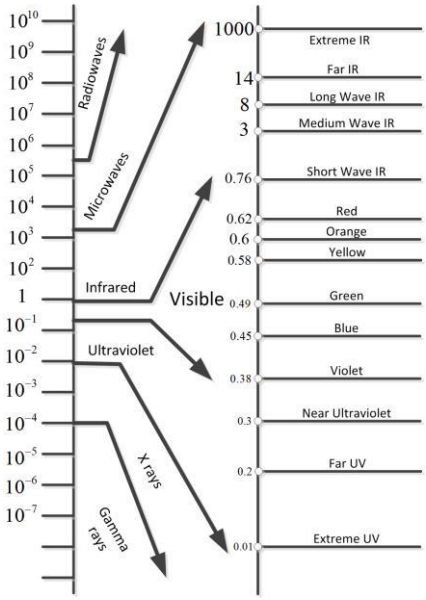


Figure 3: Main subdivisions (shades) in an extension of the visible spectrum

2.3 Interaction of radiation and matter – Spectroscopy

Spectroscopy could be defined as the use of the interaction phenomena between matter and radiation, such as

a) Absorption, emission b) Reflection c) Scattering d) Transmission

in order to provide quantitative and qualitative analysis of materials and/or study natural procedures. Matter can be seen as atoms, molecules, atomic or molecular ions or even solid-liquid-aerial objects. The interaction of radiation with matter can cause redirection of the radiation and fluctuations of the energy levels of atoms or molecules [54].

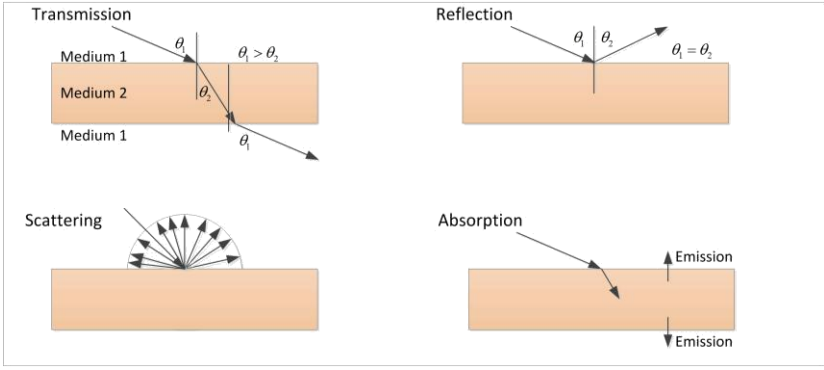


Figure 4: Any photon beam from any source that passes from a medium (usually air) and contacts with an object (2nd medium) then the phenomena of the figure will happen.

2.3.1 Absorption - Emission

The natural basis of spectroscopy is the interaction of light and matter. This was detected by Hertz during experiments discovering randomly the photoelectric effects. We consider an atomic system consisting of two energy levels E_0 and E_1 with $E_0 < E_1$ occupied by electrons. We assume that the atom is in the state $n = 0$ with energy E_0 . The atom electrons can transit to an excited state $n = 1$ of higher energy E_1 , if it interacts with radiation of energy density J (Planck's Law) of such frequency ω , so that the product $h\omega$ is equal with the energy level difference, i.e. $h\omega = E_1 - E_0$. This phenomenon is called absorption. The transition from excited state $n = 1$ in the initial state $n = 0$ with the simultaneous emission of photons with frequency ω where $h\omega = E_1 - E_0$ is called emission.

2.3.2 Scattering

The absorption of energy from a system (the scatterer) from an incident photon and the retransmission of a part of the energy of the photon from the same system to different directions, intensities, polarization and wavelength is called scattering.

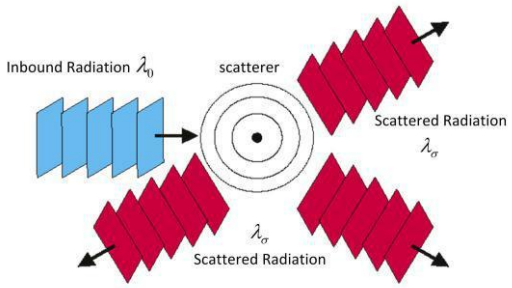


Figure 5 : The scattered radiation may have a different wavelength, intensity, phase, polarization and propagation direction of the incident radiation

The phenomenon of scattering depends on the nature of the scatterer (material, size) and its layout in space (or randomly distributed).

Types of scattering:

- Rayleigh Scattering

Rayleigh light scattering is observed for scatterers with size smaller than the wavelength λ_0 of the incident radiation. Examples of phenomena related with Rayleigh scattering are a) The blue sky color and the sky color changes during sunrise and dawn b) Energy losses in the light propagation through transparent solids (crystals, optical fibers etc.)

- Raman scattering

Raman scattering is observed during the simultaneous propagation of two photons with frequencies ω_0 and ω_s ($\omega_s < \omega_0$) through an optical medium, where the photon ω_0 is absorbed and a part is retransmitted with frequency ω_s of the accompanying photon. Energy $h(\omega_0 - \omega_s)$ is transmitted as phonon. Raman scattering is essentially an amplification of the population of ω_s photons by the photon ω_0 . Raman scattering is found in solid, liquid or aerial propagation medium and takes place when using radiation of high intensities.

- Mie Scattering

It is a powerful scattering phenomenon from objects with dimensions greater than λ_0 . There is a strong relationship between the scattered intensities and the angle of incidence.

- Stokes and Anti-Stokes Scattering

It appears during the absorption of a photon ω_0 from a molecule that causes the excitation of the molecule in a vibrating state $\nu = 1$ with the simultaneous emission of a photon ω_s where $\omega_s = \omega_0 + \omega_\nu$. Anti-Stokes scattering appears when a molecule decays from a vibrating state $\nu = 1$ to another $\nu = 0$ of lower energy with the simultaneous absorption of a photon ω_s leading to the emission of a photon $\omega_0 = \omega_s + \omega_\nu$.

- Brillouin Scattering

It appears during the propagation of a photon ω_0 through crystal medium.

2.4 Types of Spectroscopy

Spectroscopy is based on the study of the described phenomena of interaction between light and matter. Depending on the application different phenomena may present the greatest research interest. Therefore spectroscopy may focus on absorption, transmission, reflection and scattering (Raman Spectroscopy), where each kind serves different needs.

For example in diagnostic medicine while Raman spectroscopy may give more information for a sample, fluorescence spectroscopy may be preferred because the signals despite the fact that are shorter, they are more powerful and the medical exam is achieved from cheaper medical instruments. Emission spectroscopy is a little different since no external sources of radiation are needed. The reason is that the sample of study is the radiation transmitter. Typical applications of Spectroscopy can be found in various scientific fields:

- Analytical Chemistry: Spectroscopy is widely used in analytical chemistry and most of all in solutions stoichiometry.
- Biochemistry: Spectroscopy plays an important role in the area of biochemistry as it can help analyze quantitatively and qualitatively protein solutions.
- Diagnostic medical instruments: Spectroscopy is one of the basic methods for in vitro analysis since most of the diagnostic instruments use it as the main principle of operation.
- Industrial applications: Spectroscopy can be mainly used for qualitative analysis of the produced products also known as on-line control.
- Environment: One of the most important applications of spectroscopy is the measurement of pollutants in both the atmosphere and aquatic environments. Moreover Remote Sensing is an excellent tool to map and monitor areas. It is extremely useful in mineralogy and geology and it can provide valuable information regarding the climate change.

2.5 Definition of Remote Sensing

Remote Sensing is the science with the objective to acquire information about an object without making a physical contact. The term is generally used to describe the acquisition of information for Earth's surface (as well as atmosphere and sea) from aerial sensors. This is accomplished with measurement and study of the reflected or emitted radiation from specific areas of Earth and through a procedure of processing, analyzing and decision making based on this information. Remote sensing is an

application of Spectroscopy since it studies the interaction of the incident radiation with the targets under study on the surface.

Some key points of the remote sensing imaging system are the following:

A. **Source of energy and light**: A source of energy illumination providing the electromagnetic energy to the target.
B. **Interaction with the atmosphere**: As the energy travels from the source to the target, it interacts with the atmosphere.
C. **Interaction with the target** depending on the properties of radiation and the target.
D. **Measurement of the energy from the sensor**: After scattering and emission of radiation from the target, a (remote) sensor is needed to collect and record electromagnetic radiation.
E. **Transmission, reception and processing**: The energy that is recorded from the sensor must be transmitted electronically to a receiving, processing station, where data form an image and the processing stage begins.

Analysis and Interpretation: The processed image is interpreted digitally, electronically or optically in order to extract information for the nature of the target.

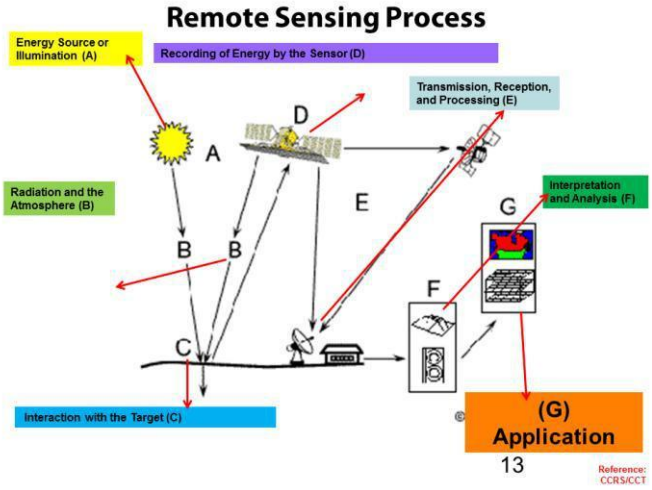


Figure 6: Schematic description of Remote Sensing [76]

2.5.1 Methods of Remote Sensing

In order for a sensor to collect and record the reflected or emitted energy from a target or a surface, it must be located to a steady platform. The Remote sensor platforms can be located on the ground (in high places), on aircrafts or on high altitude platforms in the earth atmosphere. It can also be located on a space station or satellite outside the atmosphere. Because of their orbit, satellites can continuously monitor the earth surface. It can be either geosychronous providing constant monitoring of a specific area, or low/medium orbit satellites covering a broad area

travelling around earth. Usually the choice of method for a remote sensing system is based on the cost. The main methods of remote sensing are summarized below:

- Aerophotography
- Individual Sensors (Differential Optical Absorption Spectroscopy, Thermocameras)
- Satellites

The instruments used for measuring the electromagnetic radiation are generally called radiometers and the basic categories are:

- The Imagers (image sensors)
- The Sounders (signal measurement)

2.6 Multispectral Images

Significant advances in remote sensing and sensor technology emerged from the subdivision of the broad spectrum of radiation in spectral bands of various wavelengths. This fact created the concept of multispectral images that is images of the target based on the radiation received in specific wavelengths.

A multispectral image is essentially a collection of various monochromatic images of the same object. Each of these images is taken from a different sensor which isolates a specific portion of radiation wavelength. The very well-known RGB color model for images can be seen actually as a multispectral image consisting of a red, green and blue image taken from different sensors each one sensitive to specific wavelengths.

Usually the satellites carry 3 to 7 (or lately more) radiometers. For example France's SPOT has 3 and Landsat has 7 and they receive images in various spectral bands from the visible wavelengths to far IR. Landsat 5 e.g. produces multispectral images of 7 bands where the received radiation wavelengths vary from 450 to 1250 nm [33], [65].

Remote sensing regarding Earth traditionally uses reflections in the visible and infrared wavelengths, emissions from the area of thermal infrared as well as radiowaves for the formation of multispectral images (Figure 8, [73]). The use of multiband images is also frequent for the depicturing of stellar bodies using telescopes equipped with various multispectral sensors (Figure 7, [72]).

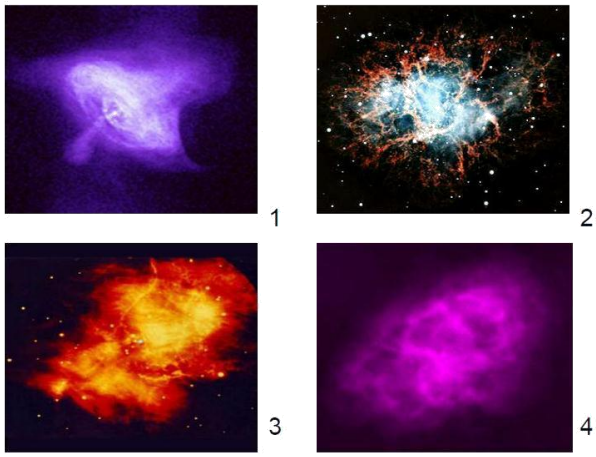


Figure 7: Four aspects of cancer Nebula (Supernova 1054) 1)X-ray band, 2)Visible spectrum 3) Infrared 4) Radio telescopic wavelengths

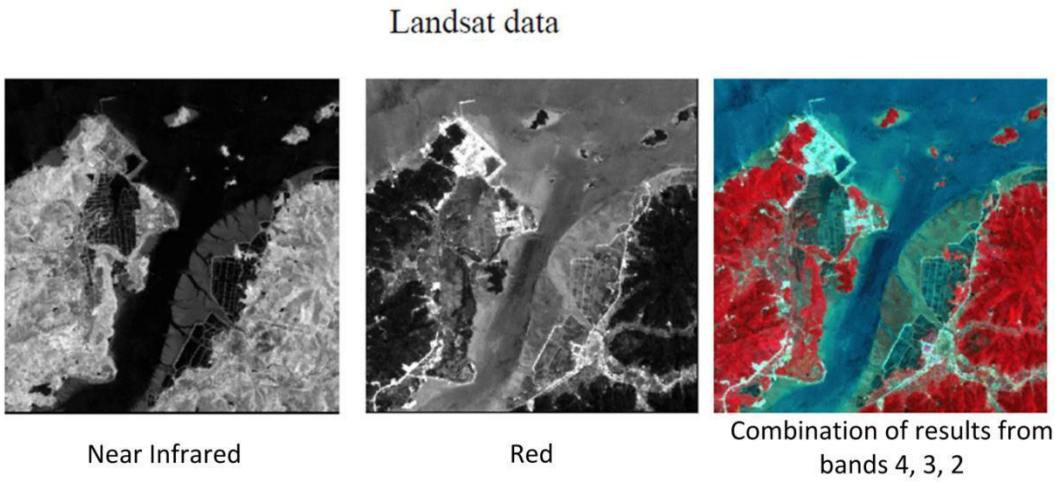


Figure 8: Images from specific bands from a Landsat measurement and a pseudochromatic image as a combination of bands 4 (infrared), 3 (red) and 2 (green)

3. Hyperspectral Imaging

The most important breakthrough in remote sensing happened with the development of the hyperspectral sensors and the proper software to analyze the hyperspectral data. Hyperspectral imaging is a relatively new technology that was originally developed mainly for the detection and identification of minerals, vegetation and features of the oceanic natural environment. Spectroscopy can be used to highlight specific spectral properties that exist due to some special chemical bonds of solid, liquid or gas materials. The detection of such materials depends highly on the size of the spectral band covered by the sensors, the bandwidth resolution of the sensors, the noise level of the radiometers and the concentration of the material on the sensor target that will define the reflected radiation power and absorption properties for the studied wavelengths. Hyperspectral data is very large datasets that require advanced and computationally efficient processing algorithms. Hyperspectral data are collected and represented in the form of a cube, the hypercube [6], [32]. The spatial information is given in *x* and *y* dimensions (as in a simple image) and the spectral information is contained in *z* dimension.

3.1 Hyperspectral Sensors

The imaging spectrometers or hyperspectral sensors are telescopic sensors that combine the spatial representation of an imaging sensor for the covered area with the analytical properties of a spectrometer.

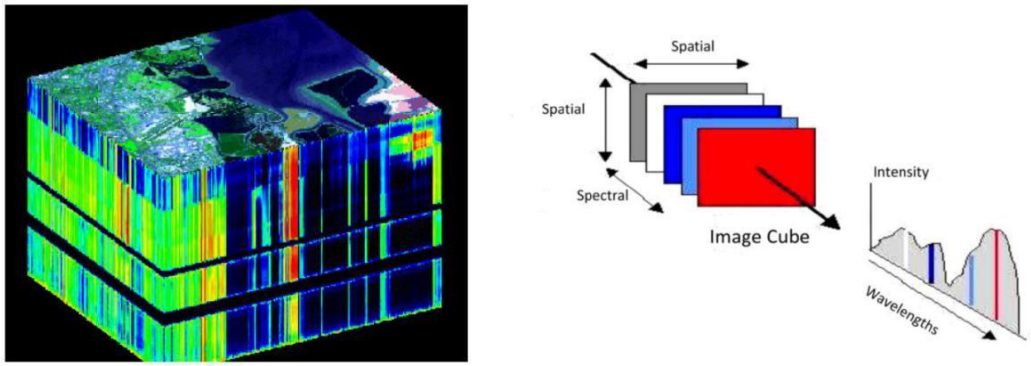


Figure 9: Hypercube example taken for AVIRIS (Airborn Visible-Infrared Imaging Spectrometer) system [32]

- They consist of many (up to hundreds) very narrow spectral band providing spectral resolution approximately 10nm or less.
- Imaging spectrometers produce a complete, almost continuous spectrum representation for each pixel of the image.

- As a result of this high spectral resolution, imaging spectrometers provide the capability of material identification and classification while on the other hand multispectral sensors could make limited distinction between materials.

Satellite Sensors	**Manufacturer**	**Number of Bands**	**Spectral Range**
FTHSI on MightySat II	Air Force Research lab http://www.af.mil/information/factsheets/factsheet.asp?id=148	256	0.35 to 1.05 μm
Hyperion on EO-1	NASA Goddard Space Flight Center http://eo1.gsfc.nasa.gov/	220	0.4 to 2.5 μm
Airborne Sensors	**Manufacturer**	**Number of Bands**	**Spectral Range**
AVIRIS (Airborne Visible Infrared Imaging Spectrometer)	NASA Jet Propulsion Lab http://aviris.jpl.nasa.gov/	224	0.4 to 2.5 μm
HYDICE (Hyperspectral Digital Imagery Collection Experiment)	Naval Research Lab	210	0.4 to 2.5 μm

Table 2: Some contemporary Hyperspectral Sensors [33], [54]

3.1.1 Spatial Resolution

For most telescopic instruments, the distance between the target that is detected and the sensor platform plays an important role in the determination of the ability of spatial information detail and resolution, as well as the size of the depicted area. Sensors on platforms far away from the main target typically can depict larger geographical areas but cannot provide detailed information. The detail observed in a hyperspectral image depends on the spatial resolution of the sensor that is defined as the smallest possible feature or object that can be detected.

The spatial resolution of a sensor depends on a factor called Instantaneous Field of View (IFOV) and corresponds to the perspective angle of the sensor. This factor defines the size of the area that can be visible from specific altitude for given time. The size of the area is equal to the product of IFOV with the distance of the sensor from the ground. This distance is also called resolution cell and it defines the maximum achievable spatial resolution of a sensor [76].

3.1.2 Spectral Resolution

Spectral Resolution is the ability of a sensor to delimit small wavelength intervals of the received radiation. The higher the spectral resolutions then the narrower are the wavelength intervals between the bands.

The sensor consists of an analysis filter bank, that is an array of band-pass filters that separate the input signal (reflected radiation) into multiple spectral components, each one carrying a specific frequency subband of the original signal. The term analysis is used to describe the process of decomposition performed by the filter bank. Thus the sensor has the ability to know the intensities of radiation per

pixel per wavelength subband. If the filter bandwidths are small then we can assume that the measured radiation for each band corresponds to a single wavelength. Low filter bandwidth on the other hand can cause other problems. The overlap between adjacent filters becomes significant and the measured correlation becomes high. This means that different bands may actually contain the same information and it cannot be used for classification purposes. Moreover the data processing workload required becomes large with no apparent gain [6].

3.2 Spectral Signatures

Reflectance spectrum shows the reflected radiation from a target object over a wavelength range. Some materials will reflect radiation of some wavelength while they will absorb radiation from other wavelength bands. These patterns of radiation reflection or absorption in a wide range of wavelengths give us the ability to uniquely identify a specific material. This is the spectral signature of the material. Some examples of spectral signatures based on studies ([45], [54], and [66]) are presented in the following figures.

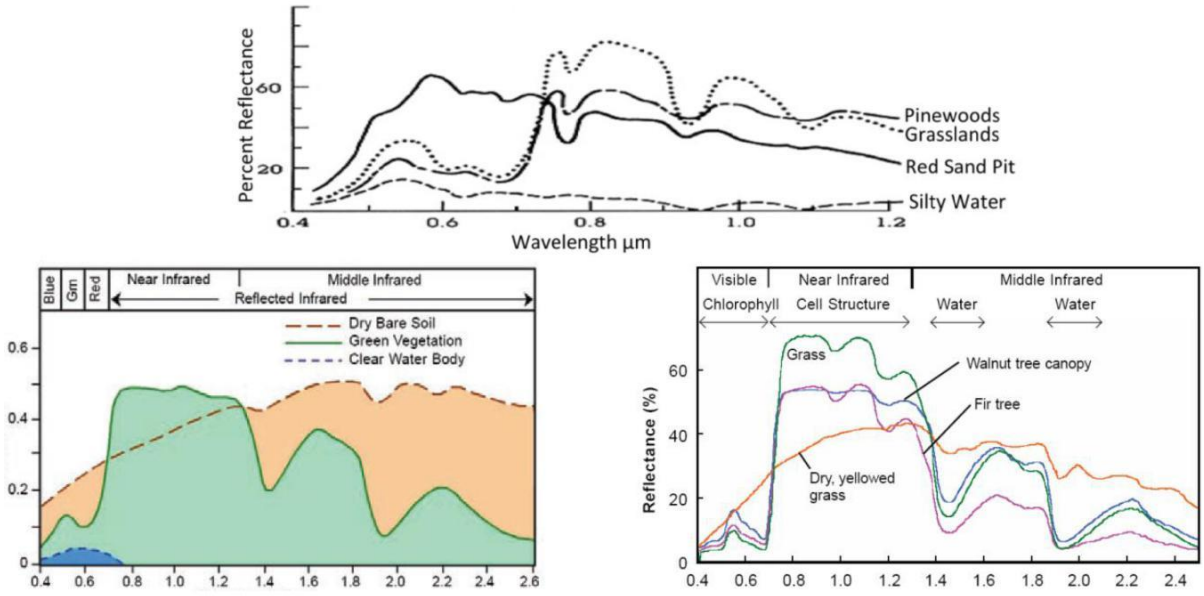


Figure 10: Spectral Signatures of various materials based on studies

3.3 Spectral Signatures

Hyper-spectral data provide us with sufficient spectral information for detection and classification of unique materials with similar spectral characteristics offering the prospect of more accurate and detailed information extraction from any other type of telescopic data.

Most multispectral sensors (e.g. Landsat, IKONOS and AVHRR) measure the reflection and/or emission of radiation in only few relatively broad wavelength bands located in non-contiguous sections of the spectrum. Therefore in many bands no actual measurement is performed. On the contrary hyperspectral sensors measure the radiation in a cascade of close and continuous wavelength bands. (Figures 11 [1], [75])

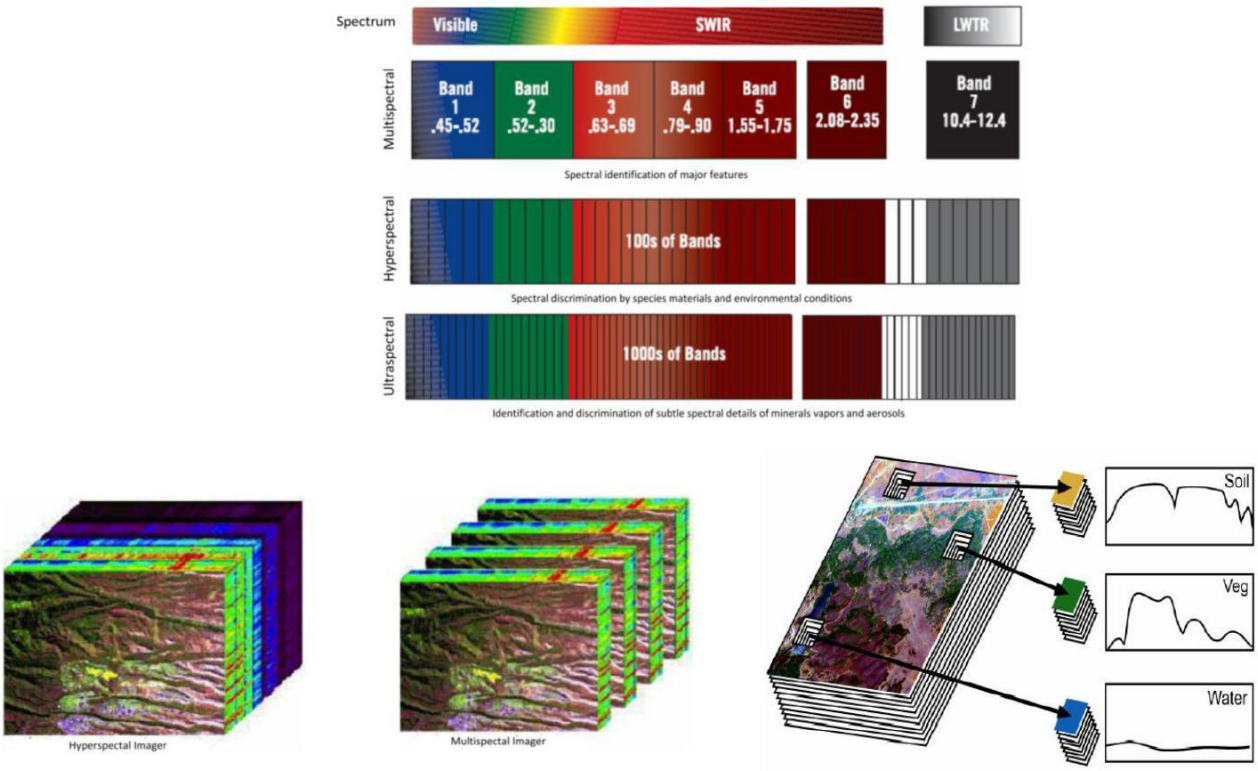


Figure 11: Spectral Imaging techniques and Comparative representation of the spectral cubes for hyper-spectral and multispectral imaging

The research spectrometers measure the reflectance in many extremely narrow bands of wavelength and therefore the calculated spectrum is presented as a continuous curve. When a spectrometer is used in a remote sensor the resulted images measure and record the reflectance spectrum for every pixel of the image. The reflectance spectrum for each pixel is very similar with the results provided by a laboratory spectrometer when measuring a specific material and therefore it provides more complete information from a similar multispectral image.

3.4 Applications of Hyperspectral Imaging

Hyper-spectral remote sensing has been used mostly by geologists to map minerals on the surface ([3], [9]) and identify soil properties such as moisture,

salinity and yields of farming crops. It is also used by agronomists and botanists to classify vegetation types [9], to study the chemical composition of plants ([31], [51]) and to identify plant diseases. Moreover it is used also by many other scientific fields including medicine with the optical biopsy as a noninvasive diagnostic method.

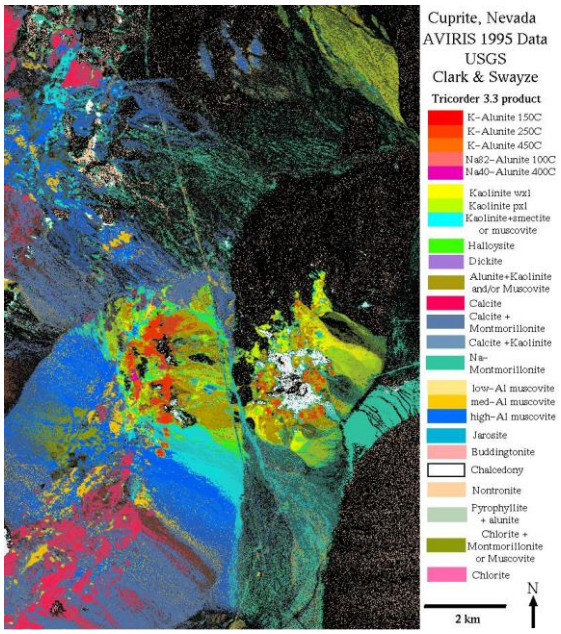


Figure 12: Example application of hyperspectral imaging in geology [75]

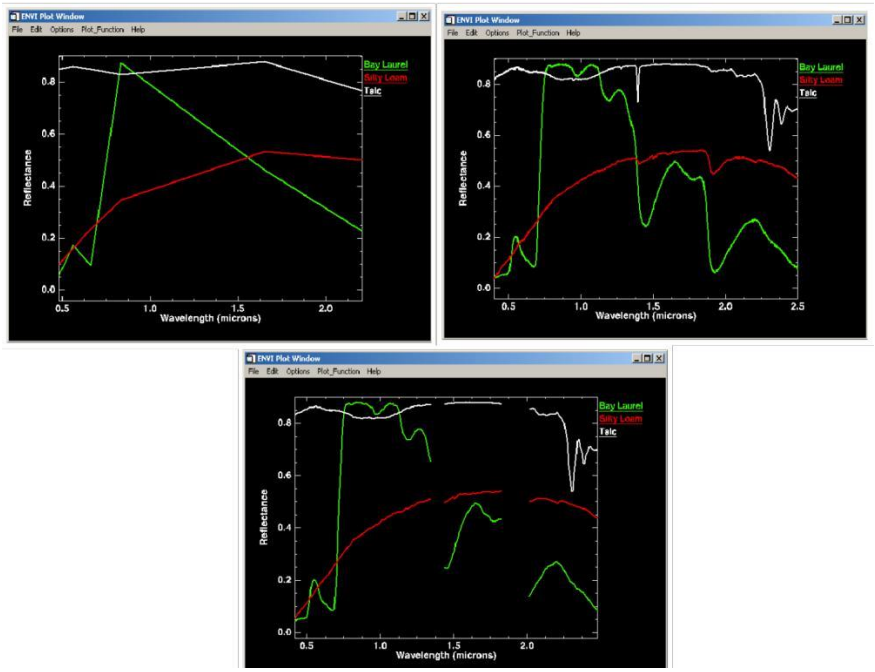


Figure 13: a) Reflectance Spectra for three materials provided by the multispectral sensor of Landsat 7 and processed by ENVI software b)Spectra for the same materials from a laboratory spectrometer c) Spectra for the 3 materials from hyper-spectral sensor AVIRIS. The curve interruptions are caused because of unreliable measurements caused by the light absorption of the atmosphere [1]

4. Processing of Hyperspectral Data

It has already been mentioned that the flow chart of a hyperspectral data processing system can be divided into three main stages: the preprocessing stage, the feature extraction stage and finally the classification. In this section we will examine some key issues related to preprocessing and study various techniques of data minimization and feature extraction before moving to the final stage of classification.

4.1 Preprocessing

Preprocessing precedes the main stage of data analysis and information retrieval. Preprocessing stage is designed in order to reduce noise, enhance some aspects from the available data, reject redundant or highly correlated information etc. Some examples are: Radiometric Calibration, Atmospheric Correction, Contrast Enhancement and Image Registration.

4.1.1 Atmospheric Correction

Before contact the Earth surface the radiation used for remote sensing travels a great distance through the atmosphere. Atmosphere consists of particles and gases that affect the received radiation through scattering and absorption mechanisms [8]. Scattering occurs when particles and large molecules of gases interact with electromagnetic radiation resulting redirection. The extent of the phenomenon depends on the wavelength of radiation, the concentration of particles and the distance covered by the radiation inside the atmosphere. Absorption occurs when atmospheric components absorb energy from the radiation at specific wavelengths. Another major problem may occur due to radiation scattering from objects adjacent to the target of interest that may present completely different reflectance characteristics. All the above mentioned issues compose one particularly difficult problem.

Remote sensing images do not contain information exclusively for the area (or object) of interest but also the atmosphere making the atmospheric correction procedure a very important piece of the preprocessing procedure. In order for us to suppress the atmospheric effects, properties and quantities such as the amount of water vapor, the composition of substances and the distribution of solid particles in the gas layers etc. must be known or estimated. However this information is not often available through direct measurements of the atmospheric features and therefore various techniques have been developed in order to export this kind of information through the hyperspectral data.

4.1.2 Geometric Correction

Due to the geometric complexity of the acquisition process, the telescopic satellite images are affected by various types of distortions and therefore they cannot be used in conjunction with topographic maps. This distortion is caused by:

- The Earth's curvature and the terrain of the land surface
- The relative motion between satellite (and aircraft) and earth during image acquisition and changes of position and velocity of the satellite in orbit.
- The spatial resolution and the angle at the time of the image acquisition.

Geometric image correction is a process that generates new images with the use of scaling, rotation, perspective projection and other transformations that give the image similar features with the topographic maps.

4.1.3 Radiometric Calibration

The ability to detect and quantify changes in the Earth's environment depends on sensors that can provide calibrated (known accuracy and precision) and consistent measurements of the Earth's surface features through time. The correct interpretation of scientific information from a global, long-term series of remote-sensing products requires the ability to discriminate between product artifacts and changes in the Earth processes being monitored. Radiometric characterization and calibration is a prerequisite for creating high-quality science data, and consequently, higher-level downstream products.

Radiometric calibration consists in linking pixels intensities to a physical parameter. Its main goal is to allow comparisons of spectra from different origins and also measurements of important physical parameters. This operation is all-important since the flux distribution in the raw spectrum is very different from what has been emitted by the observed object. Radiometric calibration contains the determination of the instrument's spectral response by the comparison of the observed spectrum with a reference spectrum for the object under study. By the division of one spectrum with the other, a calibration profile is obtained that can be used for radiometric scaling for the rest of the recorded data. It is obvious that Radiometric Calibration is connected with the atmospheric correction and the two procedures can be seen as one. However in this point Radiometric Calibration is referred to the different flux distribution coming from the radiation source (usually the sun – it is the use of standardized illuminants) and it also deals with the compensation of the effects caused by the several filters used from the sensors.

4.1.4 Data Reduction

Hyperspectral imaging data consist of hundreds of channels and together with the actual spatial size of the image can lead to large computational workload and extended processing time. Doubling the number of channels, may cause increase of the execution time by a factor quite larger than two. Moreover the simultaneous processing of highly correlated channels can cause the collapse of the classification algorithm in some cases.

In many current applications, the image sizes may be extremely big. For example, the image used during this dissertation as reference had size greater than 1 GB. Therefore it is necessary to use effective methods for feature extraction and minimization of the hyperspectral dimensions. These methods are usually called Feature Extraction and Band Reduction. If the use of the preprocessing algorithms is not done carefully, they may remove small but important differences between spectra of different materials that could be used from the classification algorithm. For example the effect of types of waste present in a vegetated area is perceived in the reflectance spectrum only in few bands and specific wavelengths so this process requires much attention.

A simple method of data reduction is based on the fact that adjacent spectral bands are usually high correlated. This way we can reduce the dimensionality of the wavelength axis without significant effect at the final results. Band Reduction is an algorithm that chooses the “best” bands as far as the classification algorithm is concerned. The choice is based in various criteria. Feature Extraction is based on mathematical and statistical models applied to data in order to achieve better expression of the original data with no correlation between zones and consequently reduce its dimensions.

4.1.4 Feature Extraction and the Feature Space

Feature extraction is the identification of features or objects of the real world into the spatial data of a hyperspectral image. The main purpose of feature extraction is, using numerical calculations, of the “best” features for specific materials. As a “good” feature we can define a quantity that has similar values for all objects/pixels of the same class (of the same material) and remarkable divergent values for objects/pixels of different classes. Usually a single feature is not sufficient to distinguish objects of different classes. As mentioned before, feature extraction is an optional preprocessing stage before classification that reduces the spatial or spectral

dimensionality of the image. It achieves by the use of spatial filters or spectral transformations to reduce the data volume and/or enhance the content of the hyperspectral data. After the feature extraction stage, it is said that the hyperspectral image has been transformed to a feature image.

For better understanding of the classification process a new term is introduced: the feature space. The satellite remote sensors measure the reflected radiation from the surface. These measurements are transformed into digital numbers, usually positive integer values from 0 to 255. Consequently each pixel of the image is characterized by a series of digital numbers, one for each spectral band. This sequence of numbers can be considered as coordinates of points in a multidimensional Cartesian space whose axes represent the spectral bands. This space is called feature space. It must be noted that this representation does not include information on the position of the pixel in the image.

To understand the usefulness of the feature space, let's consider a two dimensional example using two bands, one wavelength in the red color range and another in the near infrared range.

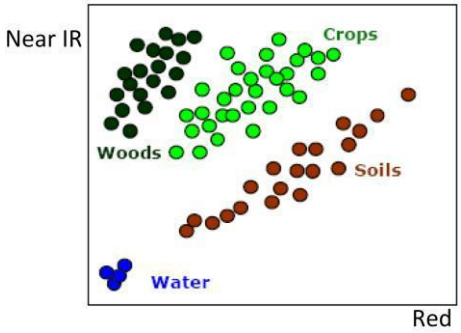


Figure 14: Two dimensional representation for two wavelengths (red, near IR) for various materials

As mentioned, objects from the same or similar materials are characterized by similar and close spectral signatures i.e. spectral reflectance responses, or else similar digital values for each spectral band. As shown in Figure 14 the pixels corresponding to the same materials are grouped together in the feature space defined from the two spectral bands. Thus the material for each group can be easily identified using these features as a spectral signature.

- The water has low reflectance rate for both red and near IR, therefore the corresponding pixels are near the origin.
- The reflectance of waste in the near IR band is slightly larger than the value of the red band and therefore the corresponding pixels are close in $y = x$ line.

- The surfaces corresponding to vegetated areas have higher reflectance in the near IR than in the red range so those pixels are located in the left part of the feature space.

The classification problem for a surface has transformed as the process of pixel assignment in one of the defined groups of the feature space. Unfortunately the representation of an image in the feature space is never as simple as in the above examples. Many pixels of an image may correspond to more than one surface due to errors, material diversity, blurring

4.2 Feature Extraction algorithms

The following algorithms are representative of the feature extraction approach that is usually used in remote sensing:

- Principal Component Analysis (PCA)
- Feedback Classification Algorithm (FCA)

During the implementation stage, it was attempted to perform data reduction with the PCA Feature Extraction technique. Code was developed to extract the transformation matrices for PCA. However due to temporal and spatial reasons we were not able to provide classification results in a data set where PCA was performed. However in order to provide a complete overview of the Hyper-spectral imaging processing chain, a brief analysis of PCA algorithm is presented and MATLAB code is provided in the Appendix.

4.2.1 Principle Component Analysis

The steps of the Principal Component Analysis are the following:

1. We calculate the covariance matrix for the given data set.
2. We calculate the eigenvalues and eigenvectors of the covariance matrix:

$$[\mathbf{V}, \mathbf{D}] = \text{eig}(\mathbf{C})$$

3. We reorder the eigenvectors relatively to the eigenvalues in a decreasing order.
4. We project the dataset on the principal components.

Calculation of the eigenvalues of a square matrix can be done via the characteristic polynomial. If we assume that λ is an eigenvalue for n×n matrix $\mathbf{A}$, then the system of linear equations $(\mathbf{A} - \lambda\mathbf{I})\,\mathbf{v} = \mathbf{0}$, where $\mathbf{I}$ is the identity matrix, has one non-zero solution $\mathbf{v}$ (an eigenvector) which is equivalent with the following expression for the

determinant: $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$. Therefore the roots of the polynomial $p(\lambda) = \det(\mathbf{A} - \lambda\mathbf{I})$ are the eigenvalues of matrix **A** and this polynomial is called characteristic. This also means that matrix **A** has at most *n* eigenvalues. Knowing the eigenvalues of a matrix **A** , we can calculate the eigenvalues by solving the equation: $(\mathbf{A} - \lambda\mathbf{I})\,\mathbf{v} = \mathbf{0}$

Principal Component Analysis (PCA) is a standardized technique for multidimensional datasets [48]. It is the process of transformation of a set of correlated variables to an uncorrelated set. The transformation is essentially a rotation of axes in various directions that are orthogonal to each other and therefore there is no correlation between the variables.

Data in hyperspectral images are represented as a matrix where the lines represent the pixels and the columns the wavelengths or spectral bands. If we study an image $n \times n$ of N spectral bands then the data matrix has n^2 lines and N columns. This is the way we calculate the covariance matrix for the specific data set. The final matrix will have dimensions $N \times N$ and will provide the relationship between the spectral bands. The orthogonal basis for a symmetric covariance matrix can be found by the calculation of the eigenvalues and eigenvectors.

The result is N eigenvalues $(\lambda_1 , \lambda_2 , \lambda_3 , ..., \lambda_{N-1}, \lambda_N)$ and the corresponding orthogonal eigenvector. If we reorder decreasingly the eigenvalues then the greatest eigenvalue will provide the direction of greater variance of data. Since the first K vectors represent the significant variation throughout the dataset we can reduce the dimensionality of the original image pixels by projecting each one to the first K vectors. Thus the dimensionality is reduced from N to K and as a result a significant reduction in the necessary computational workload is achieved. Let's assume that we have two classes and two overlapping spectral bands as shown in the figure. We also see the one-dimensional histogram of the date from the two bands. Every histogram gives one peak and it is not possible to discriminate the pixels to classes based on them. However if a new dimension is created through the direction of maximum variance then two peaks will appear in the histogram of this new "band". A second new "band" will occur orthogonal to the first. This band has small variance and the pixel samples are concentrated in a narrow range of intensities. In other words the first "band" contains most of the information and can be used for classification. The second "band" can be rejected since it cannot help us. This process reduces dimensionality from 2 to 1. The new bands are called principal components.

5. Hyperspectral Imaging Classification

Classification of hyperspectral data is the process of identification and registration of each pixel of the image in a specific class according to some statistical characteristics of the intensities values of each pixel. The hyperspectral imaging classification as a procedure can be regarded as a subset of the pattern recognition science.

5.1 Feature Extraction algorithms

Pattern Recognition is the science with the objective to identify automatically useful regularities in complicated and noisy environments. More specifically the aim of pattern recognition is to classify objectives in a number of categories or classes. A general flow diagram of a pattern recognition system is shown in the following figure:

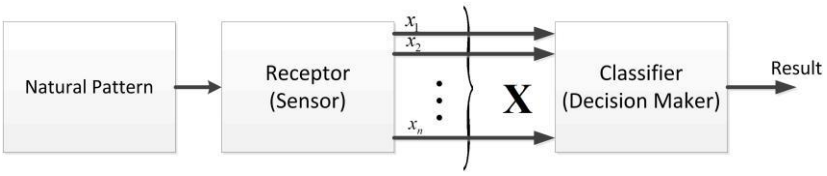


Figure 15: The general flow chart of a Pattern Recognition System

A physical pattern or model is measured by sensor measurement equipment. The sensor measures and records a series of numerical values. This set of values $\mathbf{X} = [x_1 \quad x_2 \quad \dots \quad x_n]$ is usually represented as a vector. Thus in the case of hyper-spectral imaging each pixel is represented as a vector with different intensity values in each spectral band. The vector is then fed to the classifier which decides which of the patterns is represented for each pixel [8].

5.2 Classification

As mentioned the objects are described in the feature space. In this space each object is denoted by a vector $\mathbf{x} = [x_1 \quad x_2 \quad \dots \quad x_N]$ where x_i (i =1...N) denotes one of the N features. In the case of hyperspectral imaging the objects are the pixels and the features are the intensity values for each band of wavelengths (or an equivalent transformation). The result is the classification and registration of the pixel in one of the classes ω_j where j =1...K with K the total number of distinct classes.

Formalizing the definition of the feature space in our application, we basically take the N dimensional histogram of the image and the classification is

actually the process of segmentation of the N dimensional space in K areas with the use of a decision function $f(\mathbf{x})$such as $f(\mathbf{x}) = \omega_j$, j =1...K

In any case some errors are expected to happen during classification. The decision function must be chosen so that it minimizes these classification errors. Let's see the following example:

We assume that we have a data set consisting of three typical flower species and the results are presented in the feature space, where the assumed features are the density and the length of the petals. The figure below shows that the flowers are distributed into three clearly separable clusters. For a given flower we want to create a decision rule in order to classify it in one of the predefined classes.

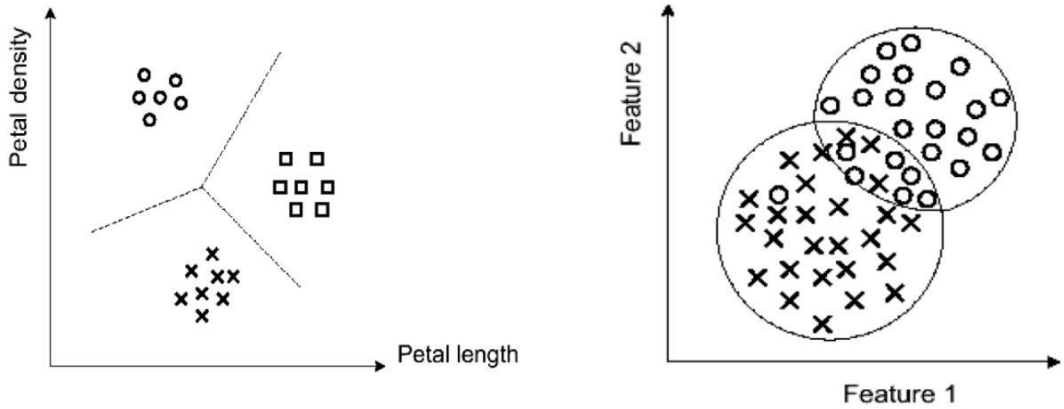


Figure 16: a) In the specific classification approach we can define specific decision boundaries in the feature space (Figure obtained by Google images) b) In this example no decision function (with no transformation) can give perfect results if we use the stochastic properties of the classes based on the minimization of mean square error [7]

5.3 Classification Algorithms

Pixels of the hyperspectral image are exported in order to train the classifier to recognize patterns. Based on these patterns the classifier creates the decision functions for the assignment of each pixel in the feature space [8]. Two main categories for classification are defined: Supervised and Unsupervised classification.

Supervised Classification: There is a prior knowledge of the spatial distribution of classes in the image. Therefore through a procedure of data collection from the depicted area, training sites are created. The spectral features of the sites are used to train the algorithm for the classification of the remaining pixels of the image. This way every pixel of the image including the training site pixels is evaluated and registered to the class that is most probable to belong to.

Unsupervised Classification: The computer algorithm automatically groups the pixels with similar statistical and spectral features (means, variances, covariance matrices) into unique clusters according to statistically defined criteria.

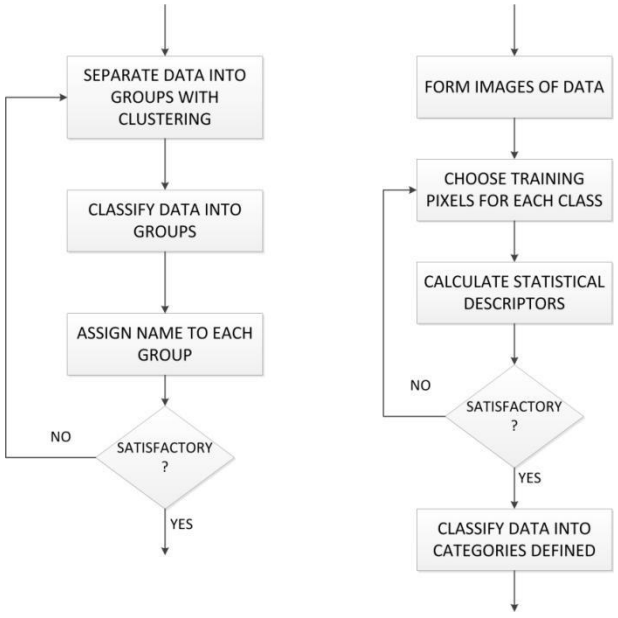


Figure 17: Block diagram of the classification algorithms (a) Unsupervised (b) Supervised

The classification methods can also be separated to parametric and non-parametric. In parametric classification the attributed of each class is defined by the PDF (Probability Density Function) of the pixels belonging to the class, so its parameters should be either known or extracted during the training procedure. An example of a PDF is the multidimensional Gaussian distribution where each dimension corresponds to a spectral band. Distribution parameters are the mean vector and the covariance matrix. Multidimensional Gaussian distribution is commonly used in Maximum Likelihood algorithm. Non-parametric classification does not use the PDF of the classes.

5.4 Training Stage

Pixel classification is performed automatically by some algorithm; however the training stage must be done carefully by the system designer/administrator. The quality of the training stage is a very important factor for classification, since it essentially defines the performance of the final result.

The main objectives of the training process are:

- To determine which classes can be identified in the image.
- To extract the statistical attributes and parameters describing the distribution of value intensities for each class.

A key requirement for the classification is the completeness of the samples of the training sites; they must include sufficient samples for all the reflective surfaces of the image. Image segmentation in reflectance surfaces is generally a slightly different procedure than the identification of areas of interest in the image. Sometimes a surface may contain pixels that present quite different values of reflectance intensities. In this case it is necessary to define smaller classes that can be identified by the designer and at the last stage by the classification algorithm. Another very significant condition for the population of the training samples is that they must be representative of the entire population of their class. This means that:

- The training sample population must be adequately large in order to provide accurate statistical estimates for the parameters of the class' distribution.
- The training samples must be distributed in an homogeneous manner in order to reflect the spatial variance of classes in the image

It is therefore concluded that there is a condition of minimum number of training samples needed for the classification algorithm that depends on the number of spectral bands used. It is generally true that the greater the number of training samples is in a class, the better statistical description will be achieved. It is nevertheless impossible to provide an analytic and accurate formula in order to calculate the needed number of training samples due to the different spatial distribution and size in each different image. In [12] it was suggested that the minimum number of samples for classification in n classes is from $10n$ to$100n$. Alternatively in [25] it was proposed that the training samples must cover a percentage between 1% and 3% of the total area of the image.

Many times the training is divided into two sets, one for calculating attributes of classes and one for certification and evaluation of the accuracy of the decision functions.

5.5 Accuracy Assessment

Classification of a hyperspectral image leads to the identification of image pixels from different materials (spectrally). When all the pixels have been assigned in one or more classes we can create thematic maps which depending on the nature of the application can be used for the identification of complete objects or the extraction of statistical attributes for these items.

When the classification of image pixels is completed, it is important to calculate the accuracy of the results. The classification accuracy is verified using a set of

samples obviously different from the one used for classification. The classified image is compared with the samples of the verification image and the result is given by an accuracy matrix [14]. The columns of the matrix represent the verification data while the lines represent the classified image

Verification Samples

Classified Image	Water	Woods	Bare Soils	Crops	Row Total
Water	**65**	4	22	24	115
Woods	6	**81**	5	8	100
Bare Soils	0	11	**85**	19	115
Crops	1	7	3	**90**	104
Col Total	75	103	115	141	**434** ←Total number of pixels

Table 3: Example of an accuracy matrix, Average Accuracy: 321/434 = 74%

Evaluation of accuracy helps us to focus on certain critical points, e.g. the poor distinction among some classes. In this case we may have to perform more preprocessing, apply different data reduction strategies or even merge these classes. In any case the entire process of classification must be repeated including the training stage.

5.6 Spectral Libraries and Reference Spectra

Spectral Libraries are collections of reflectance spectra for materials with known composition created either from observation in the nature or in the laboratory. Many researchers maintain spectral libraries for materials regarding their research field and they use them during analysis of hyperspectral images. There are many available spectral libraries (e.g. USGS Spectral Library, Johns Hopkins University Spectral Libraries etc.). The results from these libraries can be used as reference spectra for supervised classification algorithms.

The quality of reference spectra (either from measurements or a spectral library) is a very important factor with great influence in the classification results and this is why there are many methods proposed regarding the collection process. In case those unique, “pure” pixels are used as spectral images then the reflectance spectrum is almost deterministic and defined by the material of the pixel. If there are different features, materials, texture etc. in pixels of the same class then the reference spectra must be chosen from a statistic parameter. For example we can calculate the mean reflectance per band, or the median value, or the value most appeared in the dataset (for discrete set of intensities). A practical example is presented in [11]. A small test-training image area of 100 pixels is chosen for detailed analysis. Reference spectra

for this area have been extracted. The image depicts a dense forest that includes firs, shadows and rocks.

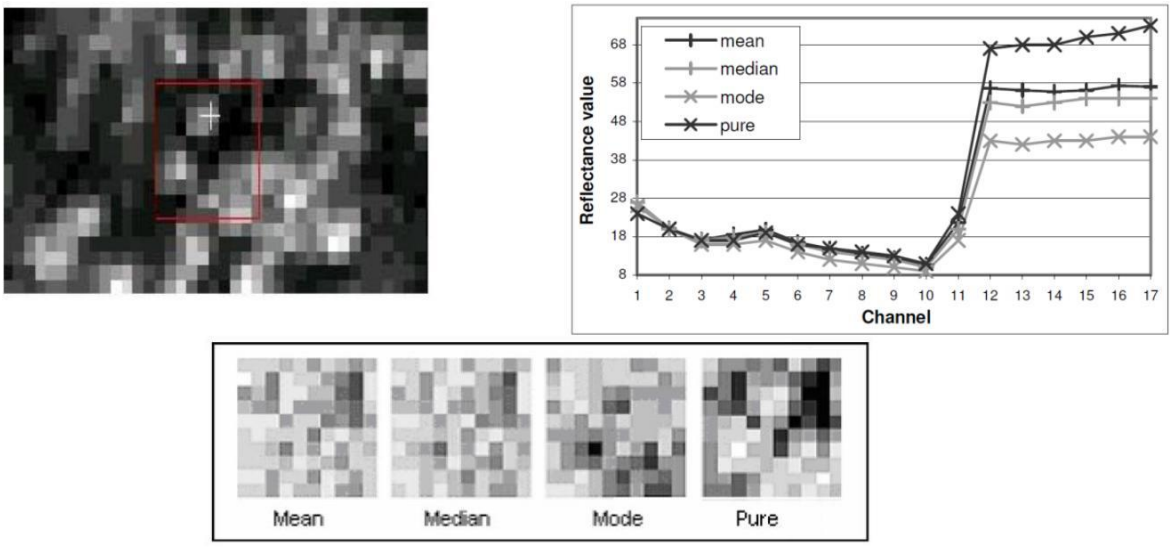


Figure 18: a) 10x10 pixel area of the hyperspectral image. The bright pixels are the illuminated tops of the trees and the dark shadows are the areas between trees b) Reference spectra have been calculated from the mean, the median and the most frequently appeared value for 17 channels. Moreover the spectrum of a "pure" fir pixel is also presented c) The spectral angles between the small test area pixels

Comparison of collection methods for reflectance spectrum has been done using spectral angles and creating representation images of the spectral angles. Thus in Figure 18(c) the bright pixels indicate the low values of spectral angle while the darker pixels indicate larger spectral angles. Low spectral angles indicate that the reference spectrum achieves better description of the actual area and therefore the population of bright pixels in the image gives us the better reference spectrum. In the presented example it was decided that mean value gives the best representation of the area and is chosen as the reference spectrum. Based on these conclusions it was decided to use mean spectrum as reference (spectral signature) during algorithm implementation.

5.7 Mixed Pixels and Spectral Unmixing

One of the biggest challenges in hyperspectral imaging applications is the process of spectral unmixing during which a mixed pixel can be reconstructed from known spectra of the image or even analyzed in separate spectra of its components (endmembers).

The existence of mixed pixels is a result of the fact that some areas are made up from different materials, features or classes that may be geometrically smaller than the sensor's spatial resolution. If we examine the hypothetical map of Figure 19 we can observe that some pixels consist of two or more classes. The extracted

reflectance spectrum in this case will be the result of synthesis of the reference spectra of each contained class. Therefore pixel identification may be extremely hard. There are in fact some methods to deal with this problem but in most of the cases there is a statistical uncertainty. A relative improvement may occur with the increase of sensor's spatial resolution (reduction of pixel size) as in the case of the small inside square of the figure reducing the number of mixed pixels.

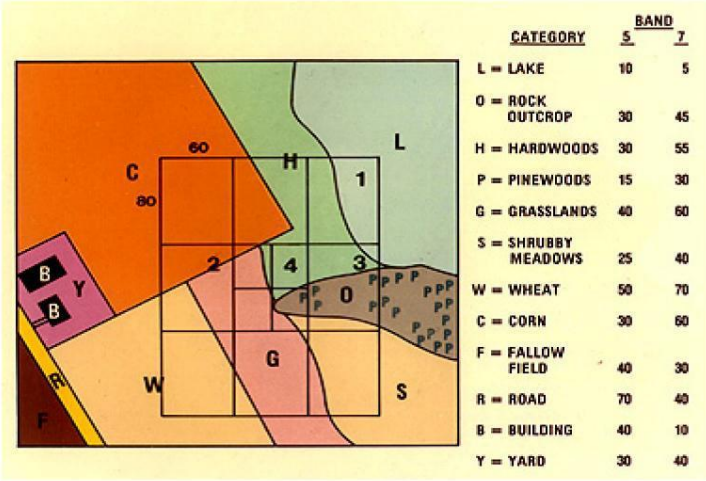


Figure 19: The rule for optimum classification regarding mixed pixels is to search for a spatial resolution that can approximate the size of the smallest class we want to identify

Spatial resolution is however limited so it may be necessary that the spectral signature of mixed pixels must be analyzed to its components so that each of the classes will be identified. It is very possible that the spectral features of a class may dominate the spectral signature of the mixed pixel. However in many applications it may be necessary to perform unmixing analysis. This term is used to define the process of detection and identification of one or more materials in a pixel. The spectrum of each pixel can then be represented as a weighted sum of the endmembers spectral signatures. Plenty of algorithms have been developed for this purpose: ICA, VCA and the FVA, (Palmadesso, et al., 1995 and Bowles, et al., 1995). These algorithms were used to approximately detect the endmembers when no prior knowledge existed regarding the materials that define the spectral mixture. In order for these algorithms to be effective, it is necessary to have access in spectral libraries that contain sufficient and appropriate samples of reflectance spectra.

During the study the effect of mixed pixels was studied but there was no attempt to perform spectral unmixing since the existing information for the area were insufficient.

Section B: Algorithms and Implementation

6. Hyperspectral Imaging Classification

Spectral classification is a subject of the Pattern Recognition science and pattern recognition algorithms can be used to identify objects and materials into predefined classes. Various classification techniques have been suggested including: minimum distance (Euclidean, Fisher Linear Discriminant, Mahalanobis, etc.), maximum likelihood and fuzzy logic algorithms such as Linear Spectral Unmixing, Mixed Tuned Matched Filtering etc. Hyperspectral data can also be classified with conventional algorithms which are often used in multispectral classification. In this case we must pay extra attention to the bands that will be chosen to extract information, since only few of the hundreds of hyperspectral bands will be used from the algorithm. However several algorithms are developed to classify hyperspectral data and they operate more effectively in the hyper-dimensional space. In these cases the training data play a prominent role in the classification procedure. In the following sections we will mention some of the basic classification algorithms for hyperspectral data [1, 25, 6, and 7].

6.1 Parallelepiped Classification

Parallelepiped algorithm uses a very simple decision rule. With the use of thresholds applied in the spectral signature of each pixel it decides if the pixel is member of a class or not. These thresholds create a *n* – dimensional parallelepiped in the data feature space. The dimensions of the parallelepiped are defined by the standard deviation of each dimension of the spectral values of each class. If the pixel value is in the range defined by the thresholds (the value is into the parallelepiped) the classifier assigns it in the specific class. If a pixel is not included in any of the parallelepipeds defined by the classes, it is assigned in class 0 and it is characterized as unclassified. Finally if the thresholds are defined in such a way that it is possible for a pixel to be included in more than one classes, it is assigned in class 255 (overlap class).

The classification accuracy for the algorithm depends on the lower and higher values for each axis. The threshold values are determined from the statistical parameters of each spectral class. It is necessary to take into account the variances of the pixels of each class before the threshold values are set.

In the example of Figure 20 it is obvious that some pixels may appear in more than one spectral box. This fact may be the result of the correlation of the spectral classes for some classes and it can be avoided with the adjustment of the parallelepiped dimensions. However smaller boxes may result in more unclassified pixels [25].

Usually the parallelepiped algorithm is used when speed is a requirement. Unfortunately this may result in cases where pixels remain unclassified or are assigned in multiple overlapping classes. The algorithm is supervised and can be either parametric or non-parametric. This relies on the designer and the possibility that statistical parameters are used for the determination of the thresholds. For example the most common rule for setting the thresholds is:

$$\mu_{c,k} - a \cdot \sigma_{c,k} \le v_k(i,j) \le \mu_{c,k} + a \cdot \sigma_{c,k}$$
$$l_{c,\min} \le v_k(i,j) \le l_{c,\max}$$

where $\mu_{c,k}$ the mean value of the pixels of class c in band k and $\sigma_{c,k}$ the corresponding standard deviation. The algorithm checks if the value $v_k(i,j)$ of pixel (i,j) is within the limits of the inequality. If the condition is true for every k then the pixel is classified as a member of class c.

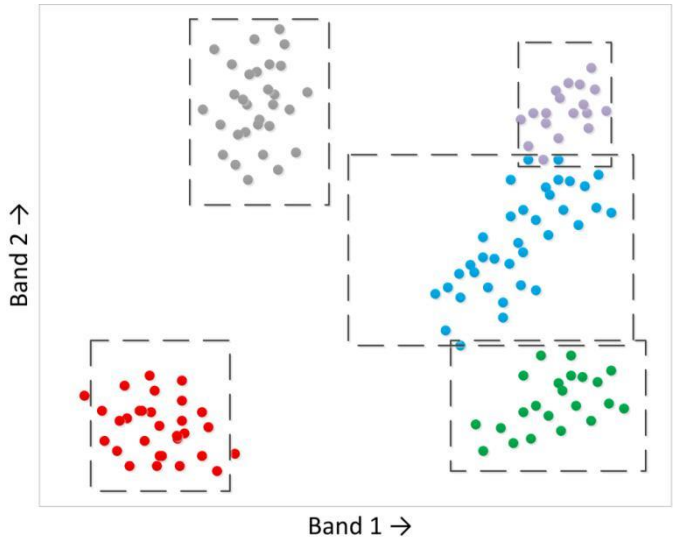


Figure 20: Example for classification with two spectral bands. Each class is defined by a spectral "box". Some parallelepipeds may overlap

6.2 Minimum Distance – Nearest Neighbor algorithms

The decision function for these algorithms is that each pixel is assigned to the cluster of the class that is nearest. Thus we need a distance measure. There are three versions of the classifier according to the definition of the nearest cluster:

1. Classify each pixel to the class whose center is nearest to the vector sample. The centers of the class are defined during the training stage. The decision function of this class may be easily implemented.
2. Classify each pixel to the class of the nearest sample that was taken into account during the training stage. Implementation of this algorithm may be quite difficult since the distance of the given sample to all the samples in all the classes must be calculated.
3. Identify the *k* nearest samples [56] that were used in the training stage and classify each pixel based on a majority rule. This version is called *k* nearest neighbor and it has the same difficulty as method 2. The two methods are the same for $k=1$

These classification algorithms are non-parametric since the decision is not based on the PDF. However the determination of the cluster centers may be defined by a statistical parameter. The algorithms may be supervised or unsupervised if an automatic clustering algorithm is used. It can also be proved that Nearest Neighbor algorithms may perform twice as worse as the maximum likelihood.

The classification results are greatly affected by the scaling of the axis in the feature space. This means that spectral bands with larger values always play a more important role than bands that may have smaller values but offer useful properties as far as classification is concerned. A common practice is to implement a dynamic range adjustment in the bands before classification, so that the intensity ranges for each band are equal. Another technique is to normalize the distance to standard deviation.

6.2.1 Euclidean Distance

In most applications it is desired that every pixel of the image is assigned to one of the predefined *C* classes. Euclidean distance algorithm takes C reference pixels of the image. These pixels represent the classes that we want to distinguish. In most of the cases the reference pixels are not actual image pixels but a statistically estimated representative pixel. This will define the center of the class. As mentioned before, the center may be the mean of the training set, or the median, or another set of values based on a custom rule. We calculate the distance of each image pixel using the Euclidean equation:

$$g_c(i,j) = \sqrt{(\mathbf{v}(i,j)-\mathbf{m}_c)^T(\mathbf{v}(i,j)-\mathbf{m}_c)} = \sqrt{\sum_{k=1}^{N}(v_k(i,j)-\mu_{c,k})^2}, \quad \begin{matrix}[\mathbf{v}_c(i,j)]_k = v_k(i,j) \\ [\mathbf{m}_c]_k = \mu_{c,k}\end{matrix}$$

where $v_k\,(i\,,\,j\,)$is the intensity value for pixel $(i\,,j\,)$ and band k and $\mu_{c\,,\,k}$ the mean value of the training pixels of class c in band k. For a given population n of the training set pixels mean value is given by:

$$\mathbf{m}_c = \frac{1}{n}\sum_{i=1}^{n} \mathbf{v}_i$$

where $\mathbf{v}_i$ are the pixels of the training set. Each pixel is registered in the class that gives the minimum distance by the following rule:

$$\text{Class}\,(i\,,j\,) = \arg\min_{c}\left(g_c\,(i\,,j\,)\right)$$

When all the pixels of the image are classified as members of a class, a new center value per class is calculated with the inclusion of the image pixels. The procedure is repeated. Iterations may continue until no pixel changes class or a predefined iteration number is reached. Usually the convergence is achieved in less than 5 iterations. The steps of the Euclidean distance classifier are:

- Get C initial means (from a training set)
- For each iteration
  - Calculate pixel distance and assign a class
  - Calculate new means based on the class members
  - If no pixel class change or a specific number of iterations is reached exit loop
- Return the classifier matrix

This algorithm is widely used but it has a major drawback. It assumes that the pixels are evenly distributed with circular symmetry around the class center and that the bands are essentially uncorrelated. It basically assumes that covariance matrix is an identity matrix. If the divergence from this hypothesis is large then the algorithm may fail.

6.2.1 Mahalanobis Distance

In order to avoid the problem induced by the Euclidean Distance algorithm, Mahalanobis Minimum Distance algorithm includes the covariance matrix in the procedure. This way the problem of non-symmetrical, correlated distributions is solved. Mahalanobis distance uses statistics of each distinct spectral class for implementation of the classifier. Distance calculation is performed by taking each class's covariance. This way we have the ability to take into account different distributions for the data classes providing highest accuracy. The distance calculation matrix is given by:

$$g_c(i,j) = (\mathbf{v}(i,j) - \mathbf{m}_c)^T \mathbf{C}_c^{-1} (\mathbf{v}(i,j) - \mathbf{m}_c)$$

where $\mathbf{C}_c^{-1}$ is the inverse of the covariance matrix for i – th class. The covariance matrix for each class is calculated from the training set of samples.

Covariance quantifies the way the pixel dimensions diverge from the mean relatively to each other. It is always measured between two different dimensions of a multivariate variable. If we calculate the covariance of a dimension with itself then the result is equal to variance. The formula to calculate covariance is similar with the variance formula.

$$\mathrm{cov}(\mathbf{X}(k), \mathbf{X}(l)) = \frac{\sum_{i=1}^{n} (X_i(k) - \bar{X}(k))(X_i(l) - \bar{X}(l))}{n-1}$$

where k , l are the dimension indexes and i is the multidimensional sample index. If the covariance between two dimensions is negative then when the value of one dimension rises then the value of the other reduces. In case covariance is zero then the two dimensions vary completely independently. If covariance is unity, then actually the two dimensions can be replaced by a single dimension since they behave with exactly the same manner. Assuming a dataset with three dimensions (x, y, z) then we can calculate $\mathrm{cov}(x, y)$, $\mathrm{cov}(x, z)$ and $\mathrm{cov}(xy, z)$. Generally for a dataset with n dimensions we can calculate $\frac{n!}{2(n-2)!}$ different types of covariance. We can therefore define the covariance matrix as follows:

$$\mathbf{C}^{n\times n} = (c_{i,j}, c_{j,i} = \mathrm{cov}(\mathrm{Dim}(i), \mathrm{Dim}(j)))$$

where $\mathbf{C}^{n\times n}$ is a $n \times n$ square matrix and $\mathrm{Dim}(i)$ is the i-th dimension. For example the element $[\mathbf{C}]_{2,3}$ of the matrix is equal to the covariance between dimensions 2 and 3. It can be easily understood that since it is a square matrix the elements in the matrix diagonal give the variances for each dimension. Moreover since $\mathrm{cov}(x, y) = \mathrm{cov}(y, x)$ the covariance matrix is also symmetric.

6.3 Maximum Likelihood and Bayesian Classifier

Maximum Likelihood classifier is based on a fundamental stochastic analysis technique and therefore basic knowledge of probability theory is required.

Conditional Probability: Conditional Probability is denotes as $p(A/B)$ and it is the probability that event *A* will occur given that event *B* is also true (has already occurred).

Bayes Theorem: If $p(A, B)$ is the joint probability that both events *A* and *B* occur, then Bayes theorem claims that:

$$p(A, B) = p(A/B)\, p(B) = p(B/A)\, p(A)$$

or else:

$$p(A/B) = p(B/A) \frac{p}{p}\left(\frac{(A)}{(B)}\right)$$

Maximum Likelihood (ML) algorithm uses the training sets to estimate mean values and the covariance matrix for each class. These estimates are used to calculate probabilities and therefore the algorithm takes into account the variations of the intensities rather than mean values. Since the estimates are produced from the training set, the reliability and accuracy of the algorithm depends on the training set collection process. If the training samples are carefully collected, then maximum likelihood algorithm performs quite satisfactory. One advantage of this algorithm is that it also provides estimates of the overlapping regions based on the statistical knowledge of distribution. It differs from the parallelepiped method that uses only the minimum and maximum values. ML algorithm is obviously a parametric algorithm for supervised classification. With proper conversions ML algorithm can be used in an unsupervised configuration.

However the algorithm does an important assumption. The intensities of the classes are assumed to follow Gaussian distribution, and the joint multivariate distribution of the vector that carries the spectral signature for the pixel follow the multivariate Gaussian distribution. Gaussian approximation can be verified from the histograms of the intensity values for the training sets. Even if there is no straight fit with Gaussian distribution, the algorithm may perform adequately. The basis of the ML algorithm is the calculation of the following probability:

$$p(\mathbf{x}_{i,j}/c) = \frac{1}{\sqrt{2\pi}\sqrt{|\mathbf{C}_c|}} \exp\left|-\frac{D^2}{2}\right|$$

where D^2 is Mahalanobis distance. $p\left(\mathbf{x}_{i,j} / c\right)$is the probability that the intensities from vector $\mathbf{x}_{i,j}$ pixel (i, j) occur given the fact that pixel (i, j) is member of class c. The decision rule for the ML classifier is based on the following rationale: For a given vector of intensities the probability that the specific pixel is member of a class is evaluated. The class that maximizes the likelihood and therefore it is more probable to be valid is selected and the pixel is assigned to the specific class. Mathematically this can be written as:

$$\text{Class}(i, j) = \arg\max_{c}\left(p\left(c / \mathbf{x}_{i,j}\right)\right)$$
$$= \arg\max_{c}\left(p\left(c / \mathbf{x}_{i,j}\right) p\left(\mathbf{x}_{i,j}\right)\right)$$

The last result is valid since $p\left(\mathbf{x}_{i,j}\right)$ is common for all the classes and does not affect the argument that maximizes the rule. If we now take into account the Bayes theorem then we conclude that the decision rule used from the ML classifier is the following:

$$\text{Class}(i, j) = \arg\max_{c}\left(p\left(\mathbf{x}_{i,j} / c\right) p(c)\right)$$

As an example, if we define two classes then if $p\left(\mathbf{x}_{i,j} / c_1\right) p(c_1) > p\left(\mathbf{x}_{i,j} / c_2\right) p(c_2)$ then pixel (i, j) is assigned in class c_1 else in class c_2. The estimates of the distributions $p\left(\mathbf{x}_{i,j} / c\right)$ are known. This is the a posteriori probability. However $p(c)$, also known as a priori probability is usually unknown. A priori probability is the knowledge of the percentage of the areas of each class that occupy the image. If we consider the case that we have no prior information then as a first step we assume that $p(c)$ is equal for all classes. Therefore ML algorithm is calculated through $p\left(\mathbf{x}_{i,j} / c\right)$. The algorithm will perform satisfactory in cases where the classes are relatively equally distributed in the image. Errors will occur if the population of one or more class is quite smaller. This problem will be dealt in a following paragraph.

Given the fact that the covariance matrix is used there is no need for axis scaling. The algorithm may also be implemented iteratively. After the classification of all pixels, mean values and the covariance matrix are re-evaluated and a new classification procedure begins. The procedure stops if there is no change in the assigned pixel classes or a predefined number of iterations are reached.

6.3.1 Bayes Classifier

Bayesian classifier is quite similar with the ML classifier but it also takes into account the cost of correct or erroneous classifications. Let's assume that L_{zy} is the cost for wrongful classification of a sample pixel that is member of class z to class y. Then L_{zz} is actually the cost of correct classification. The Classification problem can be formulated as follows. Given the probabilities $p\left(\mathbf{x}_{i,j} / c\right)$, $p\left(c\right)$ and the cost values L_{zy} then the classification for each pixel is performed by choosing the class that minimizes the cost. Total cost of a decision of classification in class y for a pixel is given by:

$$T_y = \sum_{z=1}^{K} L_{zy}\, p\left(z / \mathbf{x}_{i,j}\right)$$

where K is the total number of classes. Each term of the sum is the cost L_{zy} scaled by the probability that a pixel $\mathbf{x}_{i,j}$ belongs to class z. Decision function for pixel $\mathbf{x}_{i,j}$ is based on T_y that gives the smallest value for y =1...K. Therefore we search for the minimum value of the sum for all y. If we implement the classifier with a 0/1 cost function then:

$$L_{zy} = \begin{cases} 0 & \text{if } z = y \\ 1 & \text{if } z \neq y \end{cases}$$

This decision rule does not assign cost to correct classification and every error that may happen is regarded equally important. If we actually use this decision rule, it can easily be proved that it is the maximum likelihood classifier. However with the proper choice of costs we can make rules that take into account errors in different ways, thus the classifier gives priority to specific classes.

6.3.2 A priori Probabilities

As mentioned before the a priori probabilities $p\left(c\right)$ are not known in most of the cases, however they are needed for the ML algorithm. One method is to choose to ignore them and assume that all classes are equally distributed. This is an equivalent with a Bayesian classifier with the following cost function:

$$L_{zy} \begin{cases} = 0 & \text{if } z = y \\ \propto 1 & \text{if } z \neq y \end{cases}$$

This means that actually the cost of an erroneous classification of an erroneous

classification of a rarely appeared class is more important than errors of other classes. In many software products when this algorithm is used, it is mistakenly called ML algorithm while it is a Bayesian classifier with the cost function given above.

One other method to overcome this problem is a combination of two classification algorithms. For example Nearest Neighbor algorithms do not require a priori knowledge. It is therefore feasible to implement a system that exploits the advantages of both methods. One practical choice is to combine the two decision rules as presented in the following diagram.

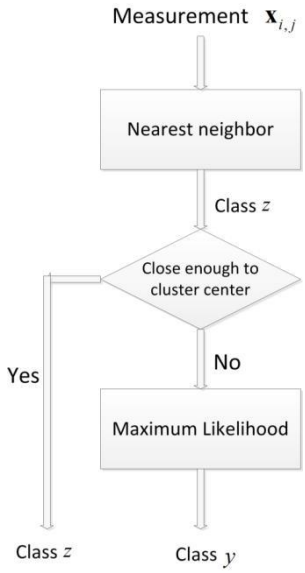


Figure 21: Combination of two classification algorithm, nearest neighbor and Maximum Likelihood

Definition of "close enough" can be made with the use of standard deviations for each class. This means that the "close enough" rule can be expressed by the following inequality:

$$| \quad | \qquad x_z - \mu_z < a\sigma_z$$

where μ_z and σ_z are the mean and standard deviation for class z. Constant α is defined by the designer and it is the degree of confidence to the decision made by the Nearest Neighbor algorithm. Moreover the results from Nearest Neighbor algorithm can be used to estimate a priori probabilities.

6.4 Null Class

Practically it is usually wise to allow the existence of a null class. The measured vectors of the null class are quite distant from the existing classes in the feature space and therefore it is not reliable to assign the corresponding pixel to any of the existing

classes. One common practice is to check if measurements in every band exceed the standard deviations of the intensities for each band multiplied by a factor.

Let's assume that the measured vector is $\mathbf{x} = [x_1 \quad x_2 \quad ... \quad x_K]^T$. We assume K spectral bands and thus feature space is K – dimensional. For each class c the mean vector is given by:

$$\boldsymbol{\mu}_c = [\mu_1 \quad \mu_2 \quad ... \quad \mu_K]_T$$

while standard deviation vector is given by:

$$\boldsymbol{\sigma}_C = [\sigma_1 \quad \sigma_2 \quad ... \quad \sigma_K]^T$$

If a vector **x** is about to be assigned to class c then it is classified to null class when:

$$|\mathbf{x} - \boldsymbol{\mu}|_c > k\boldsymbol{\sigma}, \text{ for every } c = 1...K$$

Factor k is predefined by the designer and it usually is greater than 3.

6.5 Other important classification algorithms

Spectral Angle Mapper (SAM) and Spectral Correlation Mapper (SCM):

SAM (Sample Angle Mapper) algorithm also treats each pixel as a vector in a K – dimensional space where K is the number of classes of the image. Each vector has a specific length and direction. Vector lengths represent the brightness of the target while direction its spectral features. If classification is performed based on its direction then the algorithm is not affected from brightness variations [41]. Spectral signature for each pixel is now described by the angle that is formed by the corresponding vector with the axes in the K – dimensional space. Classification is performed by comparing the spectral angles of the reflectance spectrum of the pixel under investigation with the angles of the reference spectra of all classes. SAM algorithm cannot locate the difference between positive or negative correlations since it takes into account absolute values. Spectral Correlation Mapper can solve this problem.

Fuzzy Logic Classification: In order to provide a complete picture of the available classification algorithms we will mention some popular fuzzy logic classifiers. Fuzzy Logic classifiers assign a set of probabilities for each pixel according to its tendency to belong to a class. This information can be used in a next step for a clearer definition of classes including cases where a pixel is considered mixed. There are several fuzzy logic algorithms such as: Linear Spectral Unmixing, Pixel Purity Index, Mixed Tuned Matched Filtering and Spectral Feature Fitting.

7. Implementation

The main effort for this study was the implementation of classification algorithms for hyperspectral images. Therefore several classification algorithms were implemented and applied to a test image that was provided. The objective of the study was to identify vegetated areas, urban areas, areas without vegetation (dry soil) and water surfaces (lakes, lagoons, rivers etc.). The application of such an algorithm can be of great significance:

- We can track the environmental changes and study the climate effects.
- We can prevent from observation possible environmental disasters.
- We can create a database regarding the vegetation or soil types on a given area

For example we can observe if reduction of water resources is in progress. This can lead also to reduction of vegetation and barren soils. We can also observe the rate of expansion of the urban environment or monitor new buildings appearing in the examined area.

It is clear that the importance of such a monitoring system is great for environmental purposes. However in this study we were not able to provide sophisticated algorithms that can classify different types of vegetation or land due to the fact that no external information for the given area was available. In order to implement an advanced and accurate classifier you need information about the types of vegetation that you can meet in the examined area, so that the training sets can be properly chosen. Therefore in this algorithm we tried to create basic classifiers that can isolate water resources, vegetated areas and urban or areas without vegetation. Tests were made by defining more classes (e.g. heavy vegetation – forest) but the classifier faced many difficulties due to the fact that with no a priori knowledge the training sets were chosen through visual observation and therefore the training sets were not completely accurate.

In chapter 6, five different classification algorithms were presented. However four of them were selected and implemented in MATLAB. Parallelepiped algorithm is basically an empirical algorithm that cannot provide accurate results so it was decided not to implement it. The Euclidean Minimum distance algorithm is the easiest to understand and implement comparing with the rest of algorithms and therefore it was chosen as a starting point. As it will be explained it can also be used to implement an approximation of the ML algorithm. The main objective was the study of Bayes and ML algorithms. Since both these algorithms use Mahalanobis

distance as argument in the calculated exponent, the reasonable step were to convert the implemented Euclidean algorithm in order to develop Mahalanobis Miniumum Distance algorithm and as a second step to implement the Bayesian Classifier. The ML approximation algorithm proposed is a combination of the Euclidean distance and Bayesian algorithms.

7.1 The Hyperspectral Image

The provided image is a satellite image of a rural area. There are plenty of lakes and lagoons, most of the area is vegetated, man-made buildings and constructions are limited and in some areas we can also detect streets. The image does not contain useful information throughout its complete length, but only in a diagonal rectangular. A view of the area is provided in Figure 22 after we transformed the hyperspectral image in RGB format. The features of the provided hyperspectral image are the following:

- It analyzes the reflected spectrum in 148 bands.
- The analyzed wavelengths span the electromagnetic spectrum from 428.9 nm (visible spectrum blue-violet) to 2357 nm (medium-wave infrared)
- For each band the spectrometer bandwidth is approximately 11nm. Bandwidth is expressed in FWHM (Full width at half maximum) and it indicates the spectral area where the reflectance that passes through the sensor filter is greater or equal to half its maximum value.

The image was provided in ENVI bsq format. ENVI is the most popular commercial software for analysis of remote sensing images and the files used from this software are basically the unofficially standardized file format for hyperspectral images. Therefore in most of the cases the hyperspectral images are given in ENVI file format. ENVI uses a generalized raster data format consisting of a simple flat binary file and a small associated ASCII (text) header file. This approach permits ENVI's flexible use of nearly any image format, including those with embedded header. The ENVI header file contains information used to read and understand a hyperspectral image data file. The separate ENVI text header file provides information about the dimensions of the image, any imbedded header if present in the binary file, the data format, and other pertinent information. The required information is entered in most cases automatically with the creation of the file. Especially in the case of hyperspectral images the header contains necessary information; otherwise the image cannot be exploited. Information about the bands, wavelengths and fwhm are also extracted from the header format.

Figure 22: View of the test hyperspectral image after RGB conversion

7.1.1 ENVI Header Format

The file starts with the text string ENVI and it contains some key words/properties followed by the values of these properties. Some of the important valid keywords are [77]:

Description: a character string describing the image or processing performed.

Samples: number of samples (pixels) per image line for each band. (X-axis)

Lines: number of lines per image for each band. (Y-axis)

Bands: number of bands per image file.

Header offset: refers to the number of bytes of imbedded header information present in the main binary file (for example 128 bytes for ERDAS 7.5 `.lan` files). These bytes are skipped when the ENVI file is read.

File type: refers to specific ENVI defined file types.

Data type: Parameter identifying the type of data representation, e.g. if 1=8 bit byte; 2=16-bit signed integer; 3=32-bit signed long integer; 4=32-bit floating point; etc.

Interleave: Refers to whether the data are band sequential (BSQ), band interleaved by pixel (BIP), or band interleaved by line (BIL).

Sensor type: Refers to specific sensors such as Landsat TM, SPOT, RadarSat, etc. Else the sensor is marked as unknown.

Byte order: Byte order=0 is Least Significant Byte First (LSF) data (MS-DOS systems) and byte order=1 is Most Significant Byte First (MSF) data

x start and **y start**: Parameters define the image coordinates for the upper-left hand pixel in the image.

Map info: Lists geographic coordinates information in the order of projection name (UTM), reference pixel x location in file coordinates, pixel y, pixel easting, pixel northing, x pixel size, y pixel size

Wavelength units: Text string indicating the measurement units of the wavelengths. If the wavelength units are not given, the parameter is set to Unknown.

Reflectance scale factor: The value that, when divided into your data, would scale it from 0 - 1 reflectance.

Wavelength: lists the center wavelength values of each band in an image. Units should be the same as those used for FWHM and they are set in the wavelength units' parameter.

Fwhm: Lists full-width-half-max values of each band in an image. Units should be the same as those used for wavelength and set in the wavelength units parameter. It is a measure for sensor bandwidth per band

bbl: Lists the bad band multiplier values of each band in an image, typically zero for bad bands and one for good bands.

Data gain values: Gain values for each band.

Some parts of the given header file for the specific image are presented in Figure 23. It must be emphasized that although the file was given in ENVI format, analysis and algorithm development was made with the use of MATLAB R2011a. ENVI is a commercial product but it is not as flexible or widespread as MATLAB. MATLAB provides many useful tools for efficient design of custom algorithms, it can be used as a powerful research tool and it is financially bearable.

As a first step, two MATLAB functions were created in order to automatically read information from the ENVI header file. The first function **`read_info_from_header.m`** collects all the necessary information in order to load the hyperspectral image from MATLAB. The information needed was the following:

a) number of samples per line b) number of lines c) number of bands d) possible 44

offset in the binary file e) the used byte order in the numeric format and f) the file type. The function reads the text-header file, detects the specific fields from the keywords of the file, extracts the value for each parameter and stores the value in a MATLAB variable. As far as the file type is concerned it refers to the binary file extension and it shows the method of organizing the bands of the images. The possible values are BIL (Band Interleaved by Line), BIP (Band Interleaved by Pixel) and BSQ (band sequential) which is the simpler method. In our case the file was BSQ.

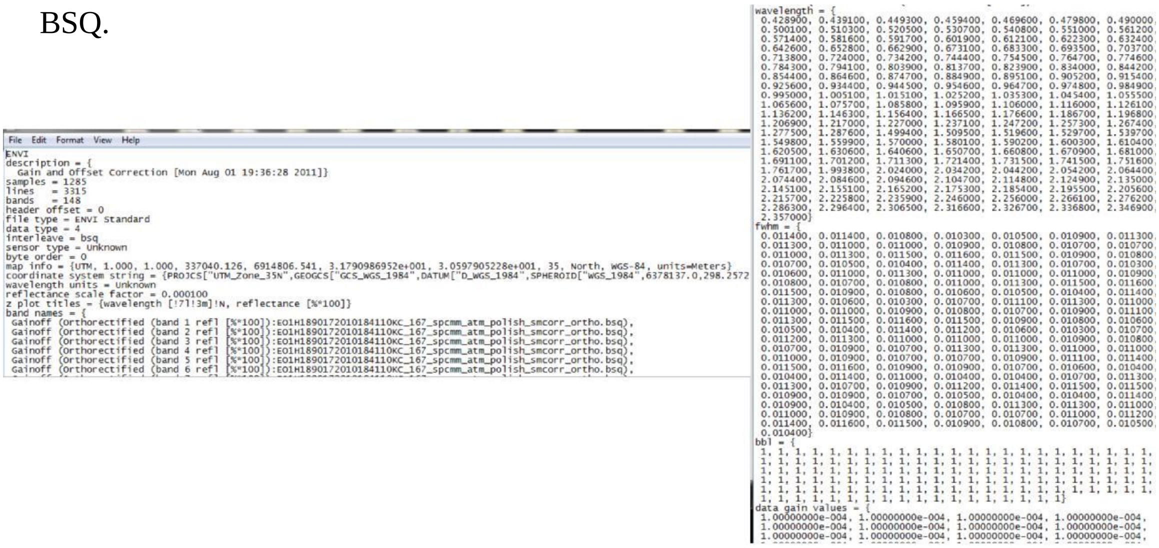


Figure 23: The header file from the test hyperspectral image

7.2 Conversion to RGB

All this information is used in order to open and process the file from MATLAB. MATLAB has an embedded function that is able to read multispectral and hyperspectral images. This function is called **`multibandread`** and it uses the image filename together with the extracted variables as input. The output is a 3D matrix with dimensions lines × samples × bands. In the case of the particular image the output of the process is a matrix with the following dimensions: 3315 ×1285 ×148. It can be seen that the size of the image may be extremely large and therefore the computer used must have plenty of available RAM to load the image. However MATLAB also allows us to load a specific subset of the image by defining special MATLAB cells as input in the function. For example the cell `{'Band','Range',[1 10]}` gives the order to load only the first 10 bands of the image. More information can be found in MATLAB Documentation.

After the image is loaded function **`read_wavelengths_per_band`** is called. This function loads the list of 148 wavelength centers for the bands of the sensor. We

also detect and read FWHM from each band as well as the normalization factor applied in the recorded intensities. The wavelength values are necessary for further processing, as well as the existence of a possible scaling factor. FWHM values were not used during this study however this information may be useful.

7.1.2 Conversion to RGB

The transformation of the image in RGB format is necessary in this study. Since we have no a priori information regarding the depicted area, the only way to identify and define the classes and set the training samples for the development of supervised classification algorithms is to perform a visual examination of the area. This cannot be done when the image is in the hyperspectral form.

Generally the visible spectrum contains wavelengths from 400 to 800nm approximately. Since the hyperspectral image contains information from the wavelength=429nm we are able to use the frequency components to convert the hyperspectral image into a format that can be used to provide an actual perspective of the area. The final goal is to express the image in RGB format.

However it is simpler to try and convert the image in a different color space. The International Commission on Illumination (CIE) has produced the CIE XYZ color space. RGB color space is reliable to accurately describe a color but the human eye does not perceive them by distinguishing primary colors. The human eye has photoreceptors (cone cells) for medium and high-brightness color vision and monochromatic night-vision receptors. In principle human eye perception can also be described by three parameters describing a color sensation. However the incoming light does not stimulate only one type of cone because the sensitivity curves overlap. An expression of the color as perceived by a human eye in tristimulus values was provided by CIE in 1931 (http://en.wikipedia.org/wiki/ CIE_1931_color_space) with the definition of XYZ color space. Y value means brightness while Z is quasi-equal to blue stimulation and X is a mixture with many red components. The CIE provided the color matching functions that give an estimate of the response of the XYZ receptors. These functions, the spectral sensitivity curves that yield the XYZ tristimulus values can also be used to provide the needed relationships that connect the intensity values per wavelength with the color space. Many different estimates of the spectral sensitivity curves have been expressed since then, while CIE in 1960 defined a simple projective transformation of the tristimulus values to the RGB color space.

From the Color & Vision Research Laboratory of UCL we found proper color matching functions that provide look-up tables connecting the wavelengths with specific XYZ values according to the estimates provided by various studies (CIE 1931, CIE 1964, Stiles and Burch). These look-up tables will provide a matching table for specific wavelengths and tristimulus values. As a first step we isolate the intensity values from the image and the center wavelengths from the bands that span the visible spectrum. Since the wavelength values from the look-up table are different from the wavelength values of the bands for the specific hyperspectral image, we perform interpolation to extract the XYZ values for the hyperspectral bands.

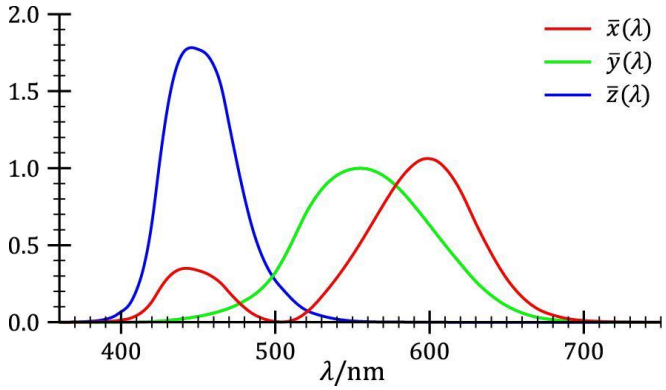


Figure 24: Spectral Sensitivity curves for XYZ color system

However the way the image is presented depends on the source of visible light that radiates to the area of interest. Each source of light has a spectral power distribution, thus it does not transmit equal energy to all wavelengths. Different picture will occur for example if the image is taken in daylight or in direct sunlight etc. Since the remote sensing images must be comparable and we cannot ensure that the lighting conditions will always be the same for every given image, we can use the standard illuminants so that the influence of the lighting source will been relatively balanced and the images can be comparable. This stage is actually the pre-processing stage of Radiometric Calibration. Since we do not have available information about the lighting conditions during the time of the measurement we can test standard illuminants that are used as reference. Standard illuminants were extracted and proposed also by CIE. CIE has proposed different kind of illuminants for different sources of light. We decided to choose the “medium daylight” illuminant with UV component also known as D65. The visual result with this specific choice was the best possible. Once more the illuminant values for specific wavelengths were found from the Color & Vision Research Laboratory of UCL in the form of look-up table.

Interpolation was once more performed to adjust the illuminant values of the look-up table to the center values of the hyperspectral bands. Since we now have all the needed information, we can perform the RGB transformation. The first step is to multiply the intensity values for each pixel with the corresponding illuminant values for the selected source of lightning. Multiplication is performed element-wise. Following MATLAB notation regarding element-wise multiplication, if we assume that a given pixel of the hyperspectral image is represented by a *num _ bands* ×1 vector then after the application of the illuminants the output is given by:

$$\mathbf{y}_{hyper} = \mathbf{I}.*\mathbf{y}$$

where **I** is the illuminant vector. As a next step the intensities of the image for each pixel and for each band are multiplied with the corresponding XYZ tristimulus values as occurred after the interpolation. If we assume that the output of the interpolation is a matrix **XYZ** with dimensions 3 ×*num _ bands* (in our case 3×148) containing all the XYZ triplets for each band and that each pixel of the hyperspectral image is described by a 148×1 vector, then the following product gives the XYZ value for the pixel:

$$\mathbf{y}_{xyz} = \mathbf{XYZ} \times \mathbf{y}_{hyper}$$

Due to the fact that the remote sensor may have performed normalization to the intensity values, the values depend on the sensor distance etc. the image that occurs may be dark. Moreover, MATLAB process images if the values of the components are restricted in the range [0,1]. Therefore as a final step we perform adaptation of the luminance of the visual target with normalization of the pixel values with the maximum recorded value of the XYZ triads of the image pixels.

The final step is to transform the XYZ image in RGB format. This function (based on the CIE standards) is implemented in MATLAB. With **`makecform.m`** function we create a MATLAB object that transforms XYZ images to RGB and with we apply the transformation to the image. The result is the image presented in Figure 22.

If we want to further brighten the RGB image we can do a very simple transformation. In RGB images brightness is controlled by the Euclidean norm of the 3D pixel vector. Therefore if we multiply it with a properly selected scalar value we can take a brighter image. This scalar factor ($a > 1$) can be found from the histogram of the image pixel norms. We can choose a value that will move the histogram to the right (to 1) but the pixel values will not exceed unity. In the specific image we cannot

find a scalar factor greater than one without exceeding unity for some pixels. However if we choose a =1.2 and set all the pixels exceeding 1 to unity then the image can be brighter with no observable distortion.

This concludes the process of image transformation in RGB format. The procedure in Matlab code can be found in the script **`convert_hyper2rgb.m`**

7.3 Data Reduction

Due to the fact that the image file size is extremely large the computer used for the implementation needs RAM memory greater than 8 GB. In order to generate code that can run in computer with less RAM memory, the image dimensions were reduced. Particularly in the presented simulation results and the provided code, the first 1000 image columns and image lines from 901 to 3150 were kept. Moreover the number of bands was decimated by 3. Data reduction was done empirically. Simple observation has shown that three adjacent bands presented very similar spectra behavior and therefore decimation by 3 will not lead to information loss. However in order to produce an accurate RGB representation of the image the bands of the visible spectrum were not decimated initially. After the creation of the RGB equivalent image, the visible spectrum wavelengths were also decimated by 3.

One of the targets of this project was to use PCA algorithm in order to reduce image dimensionality. Nevertheless due to temporal and spatial limitation this work was not concluded and it can be the target of future work. However the implemented functions that provide the PCA transformation matrices can be found in the Appendix.

7.4 Choosing Classes and Training Sets

With the creation of the RGB image we can visually examine the presented area and define the different classes. Since there is no a priori information visual examination is the only method to recognize various classes and then perform classification for every pixel of the hyperspectral image. However the resolution of the image is not high enough and we cannot observe specific details for the objects and locations presented. As a first step it was decided to attempt to define as many classes as possible. Six different classes were identified:

- Ground with vegetation
- Ground with light vegetation
- Ground with no vegetation
- Manmade construction

- Water resources
- Shallow water resources

The only information available for the characterization of the classes was the pixel colors. For example light vegetation areas had light green color and brown shades, areas without vegetation are brown or gray, water resources are dark green-blue and shallow water light green-blue color. Some manmade constructions (houses, bridges and roads) ware on the other hand logically and visually perceived. In the following figures we present areas of the classes defined at the first stage. In the rectangular frames some examples can be seen. These pixels are not unique for each class (all image pixels can be classified in one class) and the frames placed in the figures are not completely accurate (and may contain some pixels from other classes). However the reader can see and understand the class distinction. If we try to apply classification algorithms for the six different classes many problems may occur. As mentioned before each class can be characterized by its spectral signature, which is the mean value of the intensities for the hyperspectral wavelengths. If we compare the estimates of the spectral signatures for the defined classes many problems may occur. For example the distinction between areas with light or heavy vegetation cannot be simply made based on the color. Some darker green locations present similar spectral response with the light vegetation signature, which means that the darker color may be the result of darker plants rather than large densities of trees or bushes.

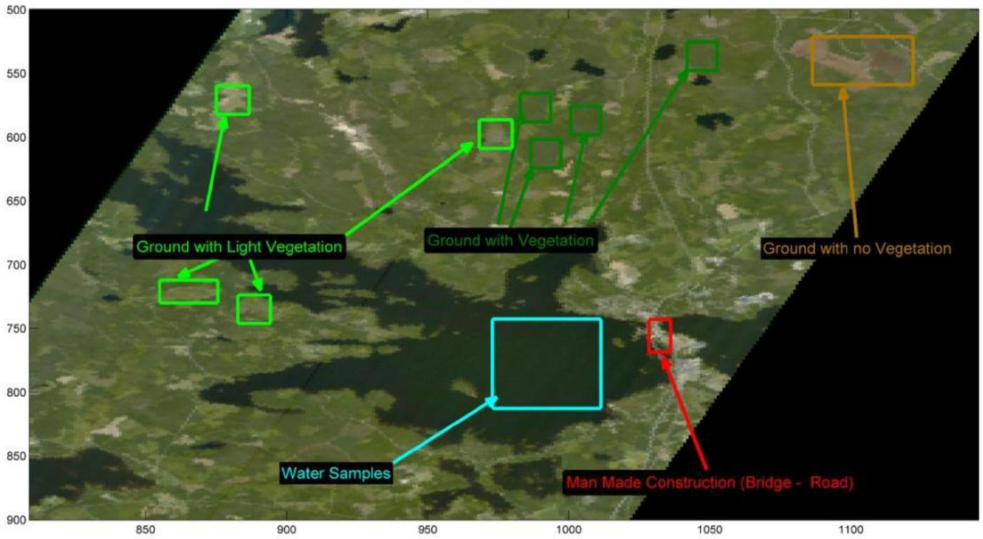


Figure 25: Example of locations containing pixels of various classes

Moreover the visual distinction between ground with no vegetation and man-

made constructions is not generally as obvious as in the examples presented in the figures. Finally the distinction between shallow and deep water ponds and lakes is not also that obvious. Thus after these observations and under the guidance of the supervising professor it was decided to merge classes and define three general and distinguishable classes:

- Water resources (shallow or deep)
- Areas with no vegetation (man-made or natural)
- Areas with vegetation

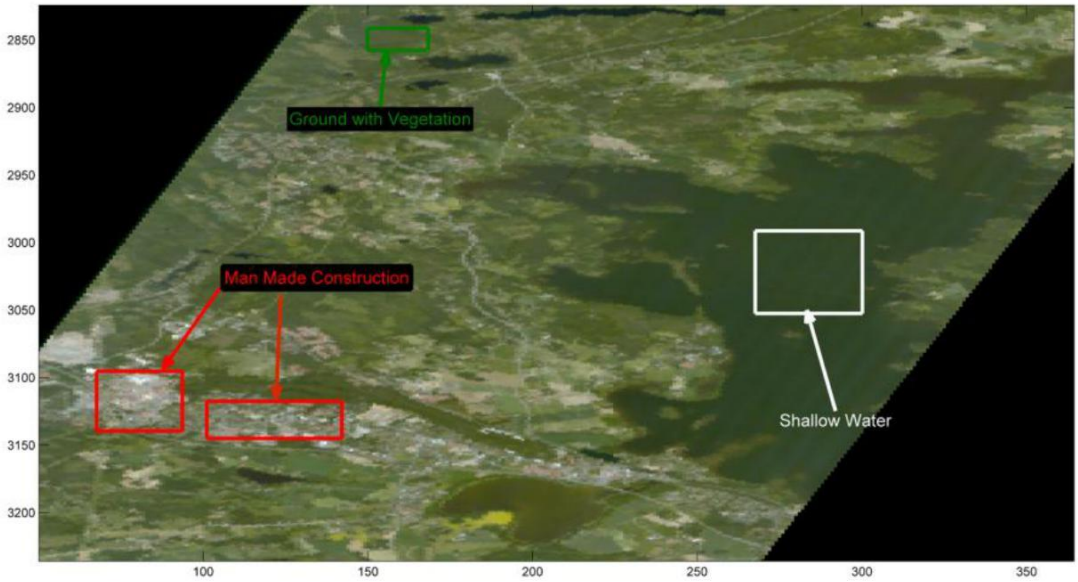


Figure 26: Example of locations containing pixels of various classes

The next step is the definition of the training sets in order to estimate the spectral signatures and the covariance matrices for each class. This is not an easy task at all. The reason lies on the fact that image resolution is quite low and many pixels are mixed. For example some pixels from the village (Figure 26) may contain trees ruining the computed estimate of the spectral signature; some brown colored pixels may contain grass or trees and the spectral signature may be similar with the one estimated for the "green" class. Therefore the pixels used for the estimation of the statistical properties of the classes must be selected so that we can be as certain as possible that they are in fact members of the classes. If the whole image is loaded (no data reduction performed!) the selected pixels of the training sets are presented in Table 4 (line and column of the image array).

It can be noticed that the sample set for the training pixels of the class that represents the areas with no vegetation is quite limited. This is due to the fact that it is quite difficult to locate "pure" pixels for the specific class since the pixels can be

easily mixed with elements and materials that belong to the class representing vegetated areas. If we include such pixels the spectral signature seems to shift to the spectral signature estimated for class C and the classifier may perform poorly.

	Pixel index - y axis (lines)		Pixel index – x axis (columns)	
	From:	To:	From:	To:
Class Water (A):	743	785	971	1000
	1091	1110	695	707
	2984	3030	272	294
Class No Vegetation (B):	3074	3085	55	62
	1941	1947	665	668
	922	928	954	956
Class Vegetation (C):	379	393	945	962
	530	539	1045	1052
	2341	2349	408	417
	508	522	966	979
	2327	2339	415	436
	187	203	1064	1077
	2182	2209	555	565
	1751	1761	640	670

Table 4: The training sets used for the test image

7.5 Estimation of the Statistical Parameters for each Class.

The next logical step is the extraction of the estimate of the spectral signature for each class. As mentioned before, spectral signature is the center of the class. It can be set following a specific rule. Usually the spectral signature is estimated as the ensemble average of the intensities of the training set for each hyperspectral band. This rule is also used in this study. Estimation of the mean value for each class will also be used in various classification algorithms. Therefore, if I_c the set containing the pixels of the training set for class c and $\mathbf{y}_{i,j}$ the intensity values for the pixel (i, j) :

$$\mathbf{s}_c = \frac{1}{N_{I_c}} \sum_{(i,j)\in I_c} \mathbf{y}_{i,j} = \hat{\mathbf{m}}_c$$

where $\hat{\mathbf{m}}_c$ is the mean value estimate for the class which is equal to the wanted spectral signature and N_{I_c} is the number of pixel in the training set. A MATLAB figure with the estimated spectral signatures is given in Figure 27.

As mentioned before the pixels for each class can be treated as random variables. Therefore we can also use the training set to estimate other parameters for the distributions of the pixels of each class. One such parameter is the variance. In order to calculate variances per band for each class the following estimation formula was used:

$$\hat{\sigma}_c^2 = \frac{1}{N_{I_c}} \left(\sum_{(i,j)\in I_c} (\mathbf{y}_{i,j})^2 \right) - (\hat{\mathbf{m}}_c)^2$$

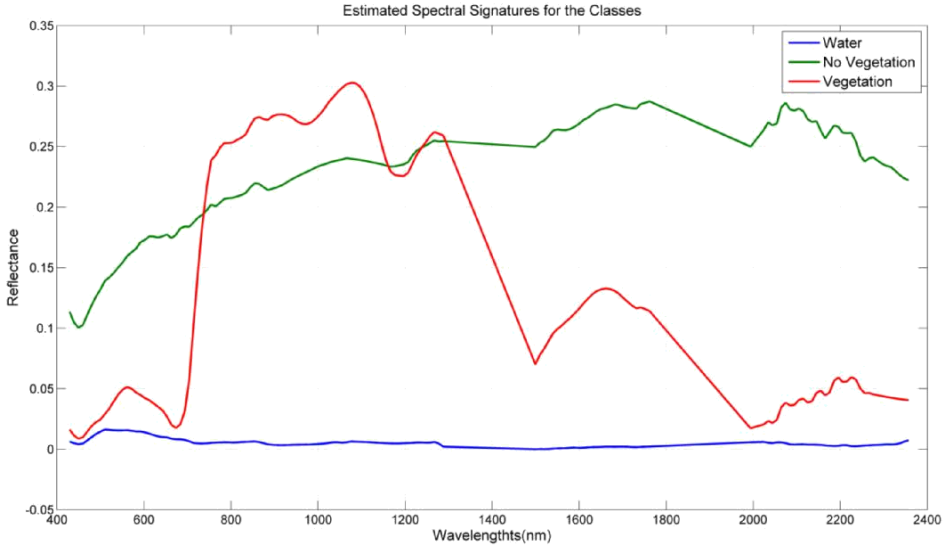


Figure 27: Estimated Spectral Signatures from the Training Set

In Figure 28 the estimated variances per class and per band are presented as plotted in MATLAB.

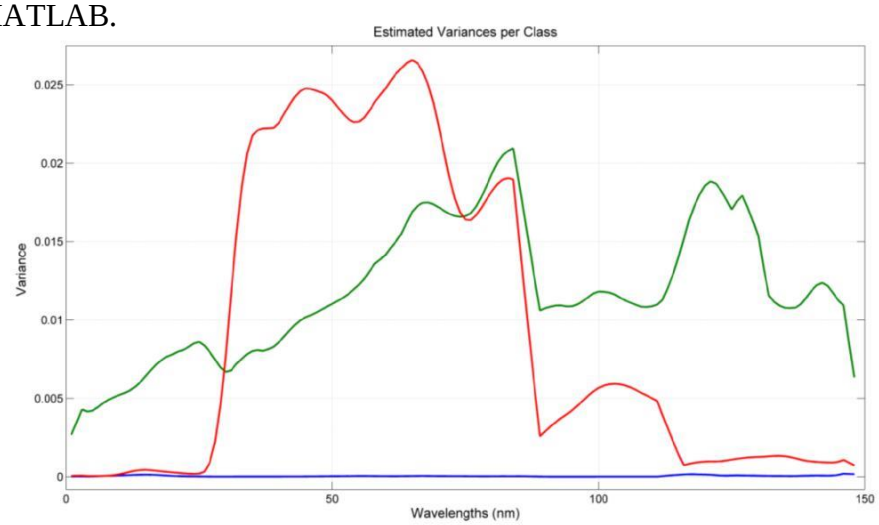


Figure 28: Estimated Variances per class and band (The colors for each curve are the same as in Fig. 27)

The common practice is to treat the pixel vectors as multivariate Gaussian random variables. In this case we need also to estimate the covariance matrix for each class. Moreover the covariance matrix is needed for various classification algorithms (Maximum Likelihood, Bayes, Mahalanobis). Estimation of the matrices is calculated from the following formula:

$$\hat{\mathbf{C}}_c = \frac{1}{N_{I_C}} \sum_{(i,j)\in I_C} y_{i,j}(y_{i,j})^T$$

The results of the Covariance Matrices for each class are presented in Figure 29 as generated in MATLAB using the mesh function. The definition of training sets and calculation of the mean and covariance matrix estimates for each class is performed

in MATLAB by the scripts **`create_spectral_signatures.m`** and **`create_cov_mtx_per_class.m`** with the use of functions **`calculate_mean_spectr_signature`** and **`calculate_cov_mtx_per_class.`**

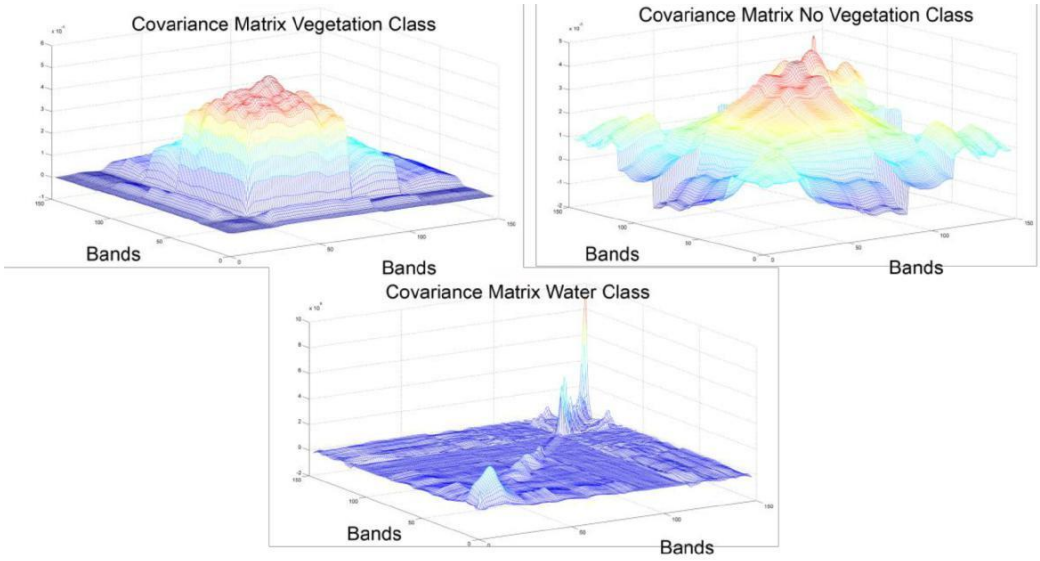


Figure 29: Covariance Matrices for each class

7.6 Classification using Euclidean Minimum Distance Algorithm

The first classification algorithm that was created and tested in MATLAB was the Minimum Distance Algorithm. In function **`euc_min_distance_class`** computation of the distances of each pixel from the class centers is performed and the class that provides the minimum distance is selected. The algorithm may run either once or iteratively. Moreover there is a choice to use (or not) scaling in order to improve performance.

Two methods of scaling were implemented in Matlab. The first method was named "maximum to one". For each band the intensities of the mean spectral signature for all classes are compared and maximum value is used to normalize the measurements of the pixel values for each band. Therefore a scaling vector (with size equal to the number of bands) is used to scale the pixels. The vector is defined as:

$$[\mathbf{s}]_k = \max\left([\mathbf{m}_0]_k, [\mathbf{m}_1]_k, ..., [\mathbf{m}_c]_k\right)$$

where $\mathbf{m}_i$ is the spectral signature for class i, c is the number of defined classes, operator $[\ ]_k$ indicates that we refer to the k – th element of the vector. Vector $\mathbf{s}$ is used to normalize pixels. Thus if $\mathbf{z}_{i,j}$ is the pixel value for (i, j) then the pixel vector used for classification is given by $\mathbf{z}'_{i,j} = \mathbf{z}_{i,j} ./ \mathbf{s}$ (element-wise division). The second

method uses standard deviation as a normalization factor for each band. If vector $\boldsymbol{\sigma}$ contains the standard deviation for each band and α is a multiplier selected from the designer then the scaled pixel is given by $\mathbf{z}'_{i,j} = \mathbf{z}_{i,j} ./ (\alpha\boldsymbol{\sigma})$. As mentioned before, scaling helps the performance of minimum distance algorithms since without it the bands where the values are smaller do not take important part in the decision process.

A second issue that must be resolved concerns the definition (or not) of a null class. If it is decided to use a null class, the value of the parameter that separates the null pixels must be set. For the specific image even though it may be desired to classify all the pixels in one of the classes, the existence of a null class seems necessary in case of multiple iterations for the following reason. The image contains many mixed pixels especially between the classes "vegetation" and "no vegetation" (areas with light vegetation). The spectral response for these pixels is in between the spectral signatures of the classes. If during the iterations we choose not to use the null class then for each algorithm run all the pixels will be assigned to one of the classes. Assuming that half of the mixed pixels are classified in class "vegetation" and the other half are classified in "no vegetation" then after recalculation of the spectral signatures of the classes including the classified sets of pixels, the estimates for the two classes will be further closer together – making classification even harder and consequently causing classification error increase. Therefore null class plays a very important role so that the mixed pixels will not be taken into account when estimating the signatures for the next iteration. Thus if it is desired to classify all the pixels of the image, the null class will be overridden only during the decision of the last iteration and the mixed pixels will not affect the spectral signatures of the classes. In Euclidean minimum distance algorithm the distance that sets a pixel in the null class is usually provided as a multiplier of the standard deviation per band. This simply means that if $\left| z_k - m_{\min,k} \right|$ is the distance of the pixel value of the k – th band from the k – th component of the spectral signature that minimized the total Euclidean distance $m_{\min,k}$, then the pixel is NOT classified in null class if:

$$\left| z_k - m_{\min,k} \right| \leq \beta\sigma_k \text{, for every } k$$

Parameter β is the null class factor that defines which pixels will be considered distant and σ_k is the standard deviation for the k – th band. It is self-evident that if scaling is used then mean values and standard deviations must be also scaled.

Finally it is emphasized once more that if the iterative approach is adopted then in each stage the spectral signatures and variances for each class must be estimated again using as training set the classification results of the previous step.

The Matlab file that implements the Euclidean minimum distance algorithm is `euc_min_distance_class.m`. In the following figures we present results for some defined scenarios. For the presentation of the classification results we created a thematic map with 4 colors. Three different colors are used for each class (blue – water, green – vegetation, yellow/brown – no vegetation) and the black color is used to depict background pixels or pixels belonging in the null class. Classification results are presented in Figures 30 and 31

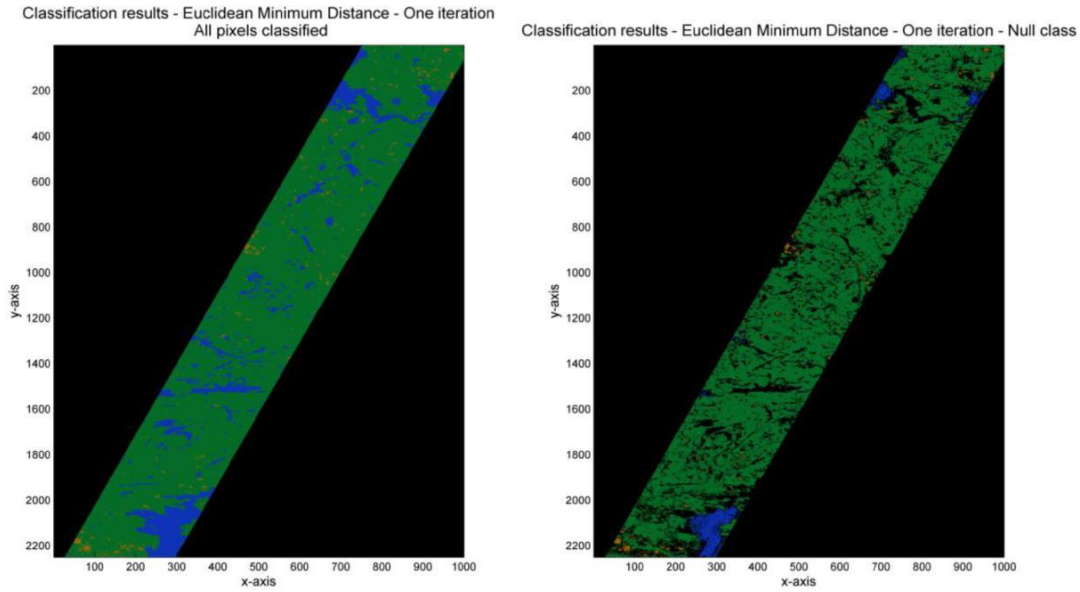


Figure 30: Classification results with the Euclidean minimum distance algorithm with one iteration (all pixels classified and with null class)

Figure 30 presents the classification results for a single run of the Euclidean minimum distance algorithm. In the first case all the pixels were classified in one of the existing classes while in the second case a null class was created to include the pixels that were distant from the estimated spectral values for all three classes. The parameter β that was selected was 3.2. It can be seen that the number of pixels classified in the null class is quite large. Moreover the algorithm seems to choose to classify mixed pixels (from "vegetation" and "no vegetation" classes) to the class with the largest training set, where the estimates of the spectral signatures and variances are quite good. Therefore the majority of mixed pixels where classified in "vegetation" class.

In Figure 31 the algorithm ran with four iterations using a null class (with

$\beta = 3.2$) for all intermediate steps. The results are quite different. The number of null class pixels at the last iteration is reduced (however the number is still remarkably large). In this case the pixels assigned in the "no vegetation" class seems closer to reality if we do a visual comparison with the equivalent RGB image. However the mixed pixels seem to be assigned mostly in the "no vegetation" class. This is caused from the fact that the number of training pixels sequentially increases. The mixed pixels contained in the training set bring the spectral signatures closer and the variance of the "no vegetation" class also increases. The result is that the majority of controversial pixels are classified as area with no vegetation.

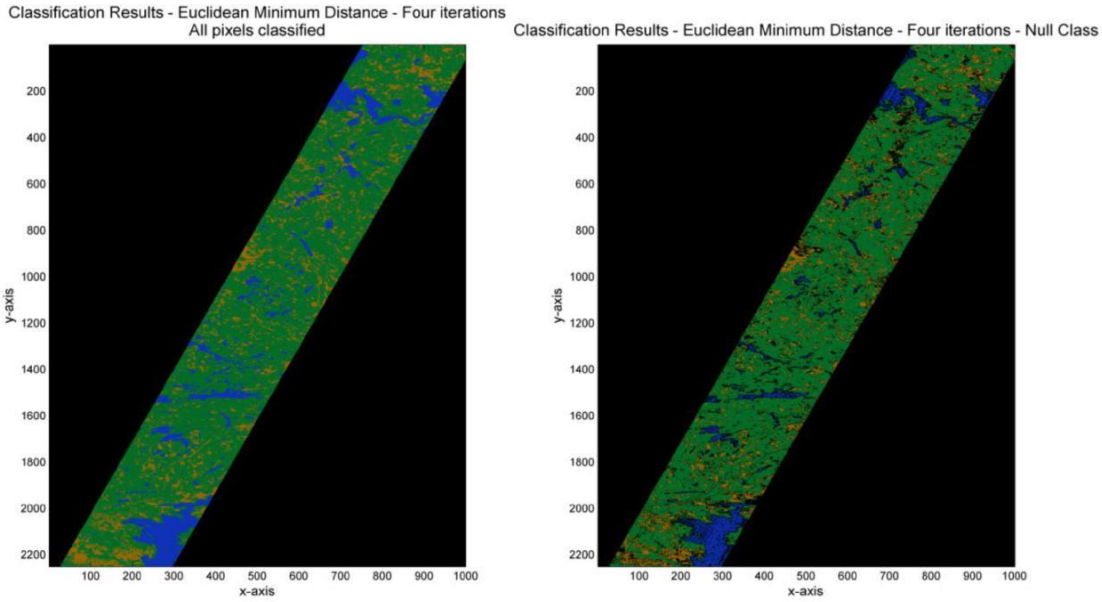


Figure 31: Classification results for Euclidean Minimum Distance Algorithm for four iterations (all pixels classified and with a null class)

The classification results are also summarized in the following matrix:

	Class "Water"	Class "Vegetation"	Class "No Vegetation"	Null Class
One iteration /all pixels	12.17%	86.24%	1.59%	-
One iteration /with null class	2.05%	67.83%	1.28%	28.84%
Four iterations /all pixels	11.98%	75.16%	12.86%	-
Four iterations /with null class	6.38%	65.45%	9.85%	18.32%

Table 5: Classification results for Euclidean Minimum Distance algorithm

Finally the spectral signatures for each case were estimated again using the identified pixels and the results are presented in Figure 32 with a MATLAB plot.

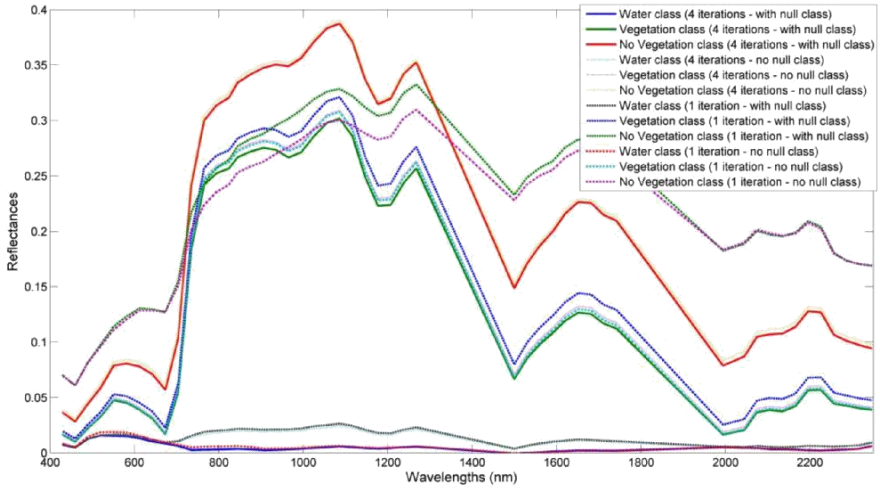


Figure 32: Estimate of the Spectral Signatures after the use of Euclidean Minimum Distance Algorithm

7.7 Classification using Mahalanobis Minimum Distance Algorithm

The second classification algorithm that was created and tested in MATLAB was the Mahalanobis Minimum Distance Algorithm. In function **`mah_min_distance_class`** an implementation of the Mahalanobis Minimum Distance algorithm is presented. The algorithm can also be performed iteratively. A significant change made concerns the calculation of the distance in order to define thresholds for the definition of the null class. As mentioned the Mahalanobis distance is calculated from the metric: $(\mathbf{z} - \mathbf{m}_i)^T \mathbf{C}_i (\mathbf{z} - \mathbf{m}_i)$. If it is assumed that the pixels of each class follow the multivariate Gaussian distribution then it can be proved that [71] *the product of the square form follows the Gamma distribution with parameters* $\frac{N}{2}$ *and* 2. The mean and variance of the gamma distribution is equal to $\mu = N$ and $\sigma^2 = 2N$ respectively. The rule for the distinction of the null class is made according to the following defined rule:

$$(\mathbf{z} - \mathbf{m}_i)^T \mathbf{C}_i (\mathbf{z} - \mathbf{m}_i) \geq N + \beta\sqrt{2N} = \mu + 3.2\sigma, \text{ for every class}$$

This rule is in some level equivalent with the rule for the Euclidean distance (using the same parameter β). In the following figures we present results for the same scenarios. The black color is used to depict background pixels or pixels belonging in the null class. Classification results are presented in Figures 33 and 34.

In Figure 33 it can be seen that when the algorithm is executed a single time then the number of null pixels occurred is quite large. Moreover the class with the smallest rate of appearance and consequently the smallest training set is generally ignored. If we decide to classify all the pixels then the number of "no vegetation"

classified pixels is increased. However, only the most representative pixels of the class are chosen. This is due to the fact that the estimate of the covariance matrix is not accurate. Moreover the pixels do not generally follow the Gaussian distribution. Thus application of the Mahalanobis metric is greatly dependent on the quality of the estimation of the covariance matrix. Small divergences may cause great errors if an estimate for a single channel is inaccurate. The problems are dealt in part with the use of iterations. This is presented in Figure 34.

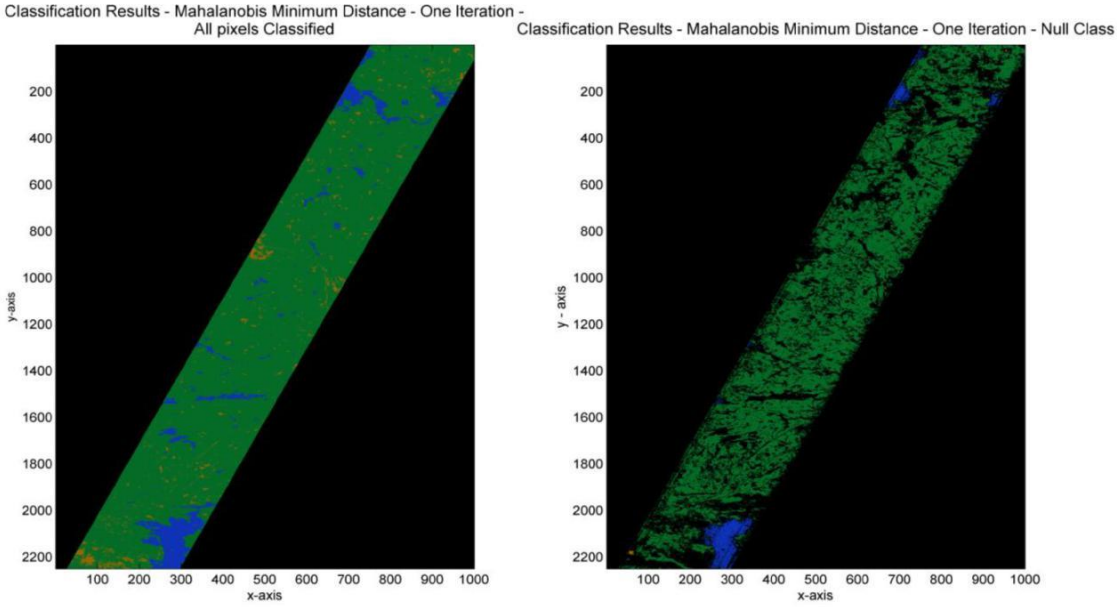


Figure 33: Classification results for Mahalanobis Minimum Distance Algorithm for one iteration (all pixels classified and with a null class)

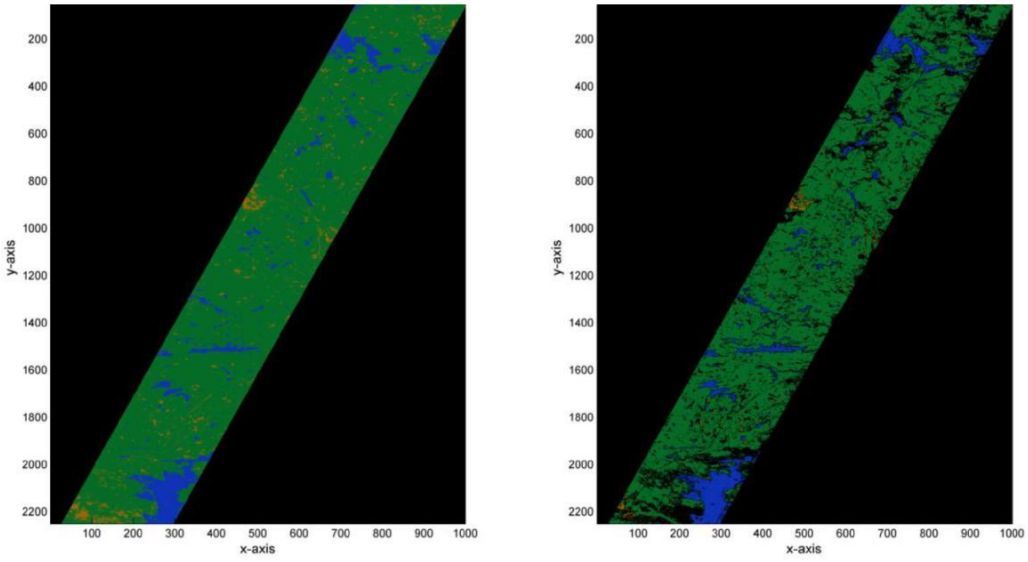


Figure 34: Classification results for Mahalanobis Minimum Distance Algorithm for four iterations (all pixels classified and with a null class)

The number of unclassified (null) pixels is radically reduced. However the number of pixels assigned in the “no vegetation class” is still quite low. The vast

majority of mixed pixels are classified to the class with the best estimation of covariance matrices and therefore the pixels are mainly identified as areas with vegetation. For example some brown colored areas in the RGB image with light vegetation are presented with green color in the classification map of Figure 34.

It must be mentioned that Mahalanobis Minimum Distance algorithm is quite slower compared with the Euclidean Minimum Distance algorithm when the MATLAB scripts run at the same computer system. This was expected since the computation of Mahalanobis Distance is performed by the product of a $N \times N$ matrix with a $N \times 1$vector from the left (transposed vector) and right. The algorithm is also slower after a single iteration is completed since it is required to calculate a new estimate not only for the spectral signatures (common step with the Euclidean Distance) but also for the covariance matrix. However the cost paid in computational complexity does not seem to provide any profit in the case of this image where a class ("no vegetation") consists of a small number of pixels and the estimations of the statistical parameters cannot be accurate.

The classification results are also summarized in the following matrix:

	Class "Water"	Class "Vegetation"	Class "No- Vegetation"	Null Class
One iteration /all pixels	6.52 %	90.58%	2.91%	-
One iteration /with null class	2.44%	55.42%	0.04%	42.1%
Four iterations /all pixels	7.94%	88.03%	4.03%	-
Four iterations /with null class	2.47%	67.88%	0.92%	24.47%

Table 6: Classification results for Mahalanobis Minimum Distance algorithm

Finally the signatures for each case were estimated again using the identified pixels

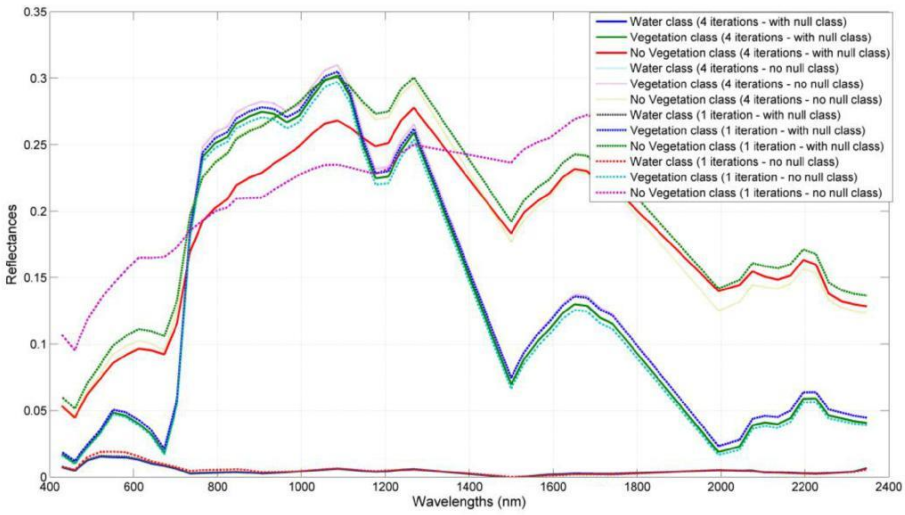


Figure 35: Estimates of the Spectral Signatures after the use of Mahalanobis Distance Algorithm

7.8 Classification using Bayes Classification Algorithm

It must be clarified that since the term “Bayesian Estimator” can be applied in many cases, the Bayes Classification Algorithm that is implemented is based on the description given in paragraph 6.3.1. This algorithm therefore performs classification based on estimations of conditional probabilities without any knowledge of the a priori probabilities which express the incidence of a pixel of each class. As mentioned in 6.3.1, this algorithm assumes that the a priori probabilities are all equal for all classes and therefore classification is performed based on the conditional probability that a given pixel is a member of a class. The pixel is assigned to the class that maximizes this probability. In function **`bayes_class`** an implementation of the Bayesian classification algorithm is presented.

It was made clear that both the assumptions that were made previously are not quite accurate. The class “vegetation” in the specific image is dominant and therefore the actual a priori probabilities are quite unequal. Particularly the probability of appearance of a pixel from “no-vegetation” class is very small. Moreover the Bayesian algorithm assumes multivariate Gaussian distributions which are not also very accurate. However from the results presented in Figures 36 and 37 it is concluded that the algorithm performs extremely well if we visually compare the color classification maps and the RGB image.

As far as the null class is concerned we can use threshold probability values to separate pixels that are quite distant from the spectral signatures of reference from all the classes. For the presented simulation results this threshold was chosen to be $\gamma = 0.15$. This means that if a pixel cannot be assigned to a class with probability greater than 15% (based on the assumed distributions) then the pixel is classified as null. The rule that defines which pixel is assigned to the null class is described by:

$$p\left(i / \mathbf{z}\right) < \gamma, \text{ for every class } i$$

The results presented in Figure 36 indicate a great improvement in comparison with the two previous classification algorithms as far as the reduction of the null class pixels. A very small number of pixels are unclassified after a single run of the algorithm. However a closer look can reveal that if we choose to classify all pixels in one of the three defined classes’ then some clear classification errors may occur. For example in the area around pixels (x = 450, y =950) the set of pixels that are classified as null in the right subfigure are wrongfully classified as “water” pixels if it is decided to assign all pixels in a class. This error is a consequence of the fact

that distribution parameters are inaccurate for a small training set and that the a priori probabilities are unknown. This problem is however avoided if more iteration is performed.

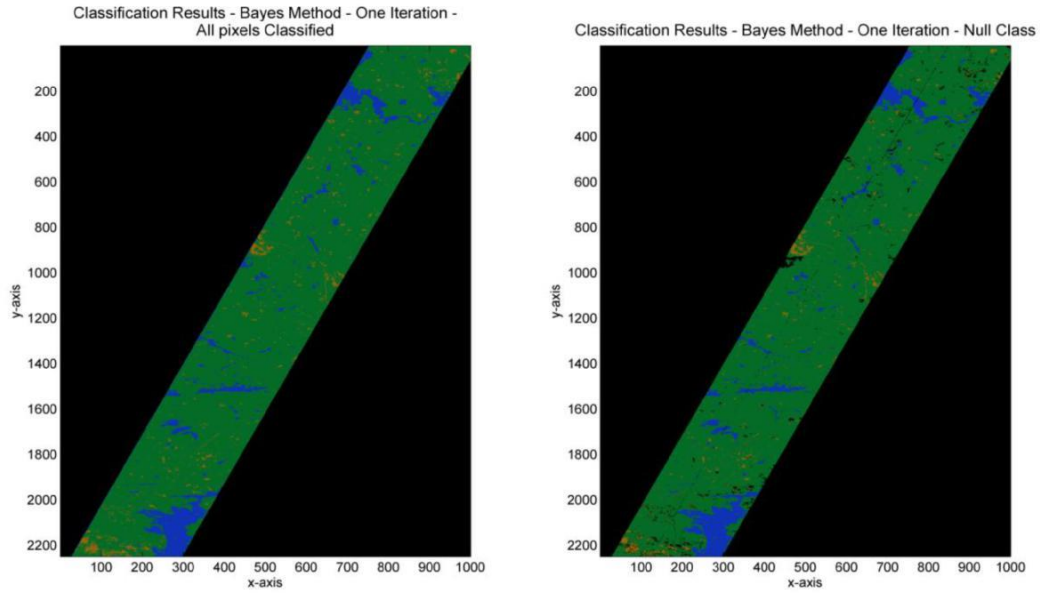


Figure 36: Classification Results for the Bayesian Classification with one iteration (all pixels classified and <u>with null class)</u>

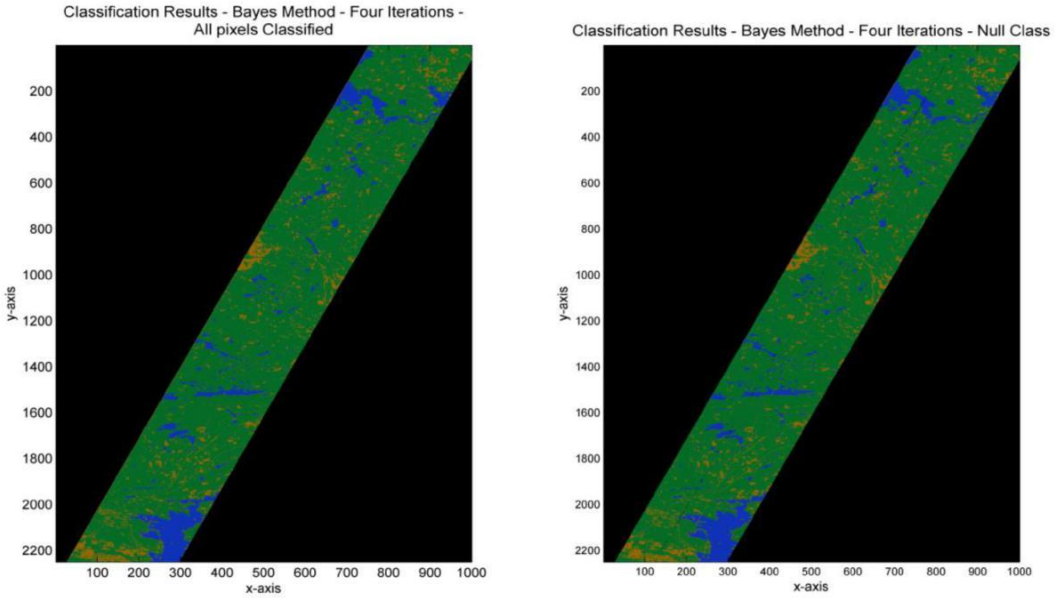


Figure 37: Classification Results for the Bayesian Classification with four iterations (all pixels classified and with null class)

As presented in Figure 37 the algorithm seems to work exceptionally well. The number of null class pixels is reduced even more and the classification result seems to be extremely accurate with a visual comparison of the color map with the actual RGB image. The algorithm seems to have no bias towards a class as far as the mixed pixels are concerned. The mixed pixels are fairly divided between "no-vegetation" and "vegetation" classes and the color map is actually a very good

representation of the actual image. The number of "no-vegetation" pixels is increased with iterations as expected but not so much that it would lead to erroneous classification. The pixels of the "water" class are also very accurately determined. However a small bias towards to the class with the smallest a priori probability can be observed.

Since the calculation of the conditional probability demands the calculation of the Mahalanobis distance it is also clear that this algorithm is slower than the Euclidean Distance Algorithm. The additional complexity comparing with the Mahalanobis algorithm (calculation of the exponents) is very small and therefore the two algorithms have almost the same performance. Moreover the Bayes classification algorithm has no gain from the scaling procedure and therefore we can save computational time if the scaling step is disregarded.

The classification results are also summarized in the following matrix:

	Class "Water"	Class "Vegetation"	Class "No- Vegetation"	Null Class
One iteration /all pixels	8.33%	88.82%	2.85%	-
One iteration /with null class	7.65%	87.49%	2.35%	2.52%
Four iterations /all pixels	9%	81.72%	9.28%	-
Four iterations /with null class	8.93%	81.68%	8.92%	0.47%

Table 7: Classification results for Bayes classification algorithm

Finally the spectral signatures for each case were estimated again using the identified pixels:

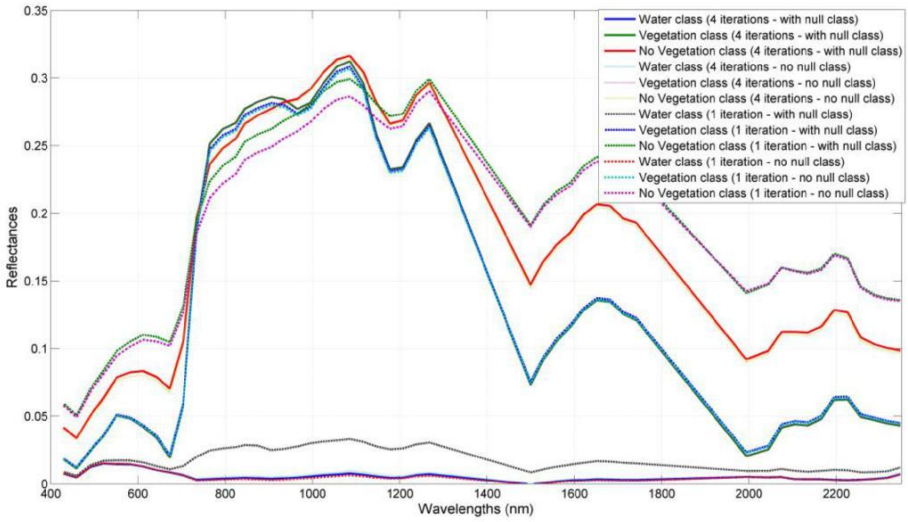


Figure 38: Estimates of Spectral Signatures after the use of a Bayesian Classifier

7.9 Classification using an ML Approximation Algorithm

The last algorithm that was implemented in MATLAB was an approximation of the Maximum Likelihood classification algorithm. In paragraph 6.3 it was highlighted that in order to use the ML algorithm, the a priori probabilities must be known. Since it is not possible to know a priori probabilities in most of the cases (such as in the test image used), an approximation of the ML algorithm was created. This was done in two basic steps:

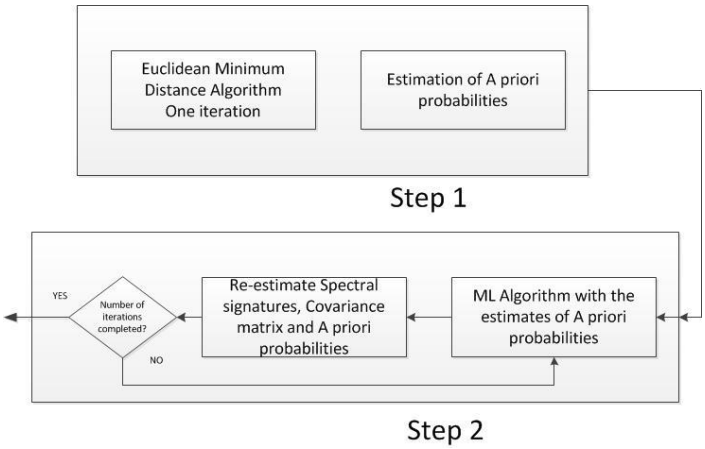


Figure 39: The ML Approximation Algorithm

The first algorithm used is the Euclidean Minimum Distance algorithm with a single iteration. Null class was used and the a priori probabilities were estimated based on the following formula:

$$\hat{p}(i) = \frac{\text{number of pixels classified in } i \text{ class}}{\text{number of pixels classified in any class (except null)}}$$

Euclidean Minimum Distance algorithm is fast and it can provide a relatively realistic approximation of the a priori probabilities. In any case the estimates are more reliable than the use of equal probabilities as in Bayes Classification.

Then the ML Algorithm was used (implemented in `mle_class` function) to definitively assign the pixels in classes. The use of iterations was also provided. For each iteration the a-priori probabilities were calculated along with the needed statistical parameters. The algorithm is the same with the Bayes classifier with the only difference that instead of using $p(i/\mathbf{z})$ to make the decision $p(i/\mathbf{z})\,\hat{p}(i)$ is used. The same change is applied also to the rule that defines the null class. The results are presented in Figures 40 and 41.

It is clear that there is improvement in any case. With a single run almost all the null class pixels can be eliminated, no significant classification errors are made

and if we decide to assign all the pixels in one of the classes, the result is very satisfactory. The classification results are improved with the use of iteration in the second step of the algorithm. A visual comparison to the RGB image shows that the results are exceptionally accurate. The number of "no-vegetation" areas is slightly reduced comparing with the Bayesian classifier due to the use of the a priori probabilities but in this case this fact is an improvement since the Bayesian classifier even though it is quite fair, it has a small bias to the rarely appearing classes.

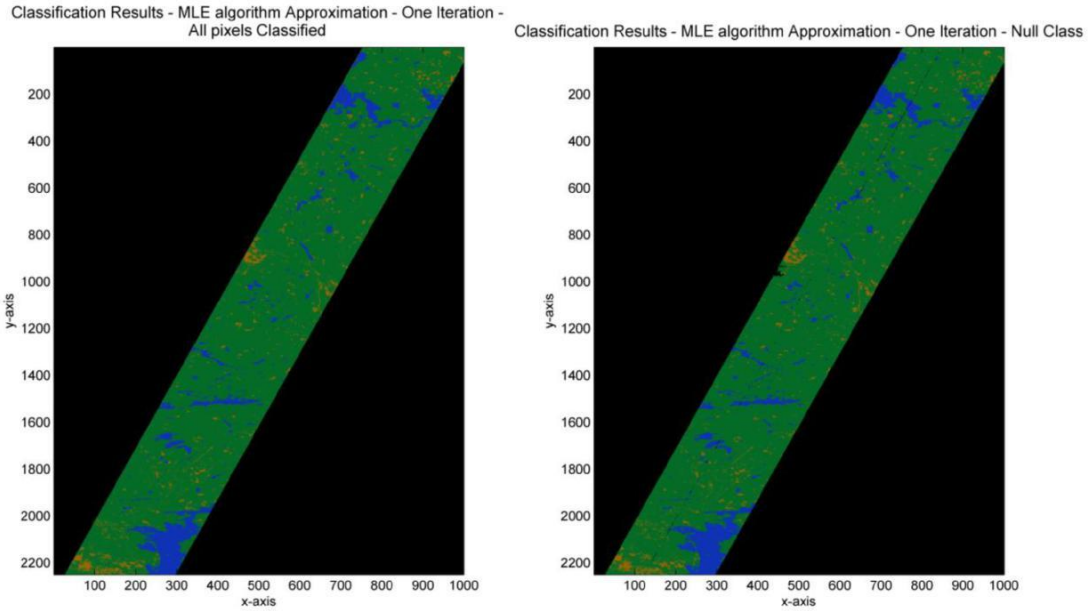


Figure 40: Classification Results for the ML Approximation with one iteration (all pixels classified and with null class)

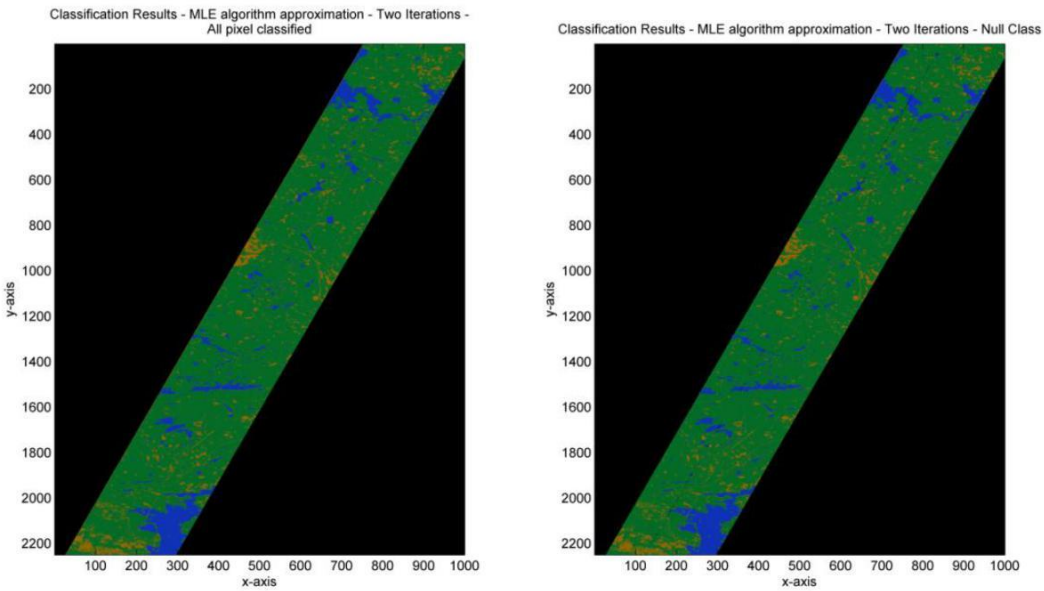


Figure 41: Classification Results for the ML Approximation with two iterations (all pixels classified and with null class

It must be emphasized that we only used two iterations (instead of 4) to achieve this

result. The classification results are also summarized in the following matrix:

	Class "Water"	Class "Vegetation"	Class "No- Vegetation"	Null Class
One iteration /all pixels	8.25%	88.91%	2.83%	-
One iteration /with null class	7.58%	87.58%	2.32%	2.53%
Two iterations /all pixels	8.56%	87.38%	4.06%	-
Four iterations /with null class	8.39%	87%	4.01%	0.59%

Table 8: Classification results for ML approximation algorithm

Finally the spectral signatures for each case were estimated again using the identified pixels:

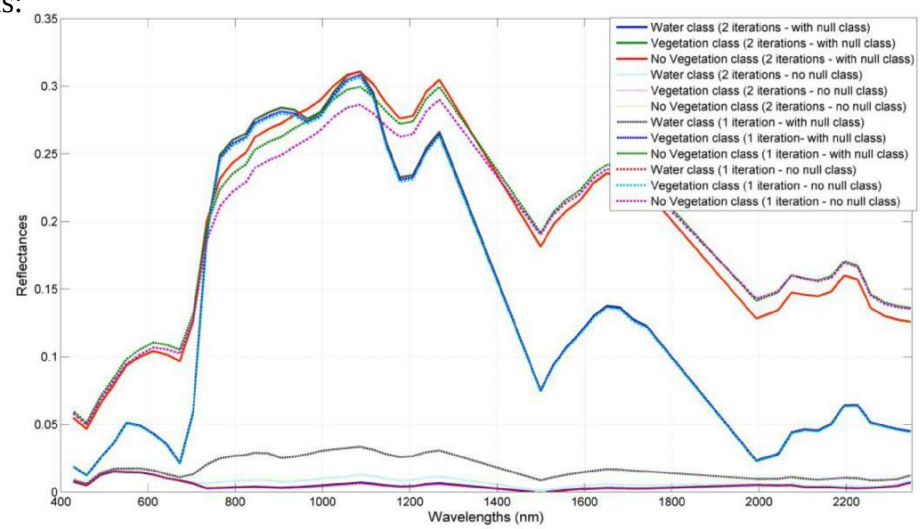
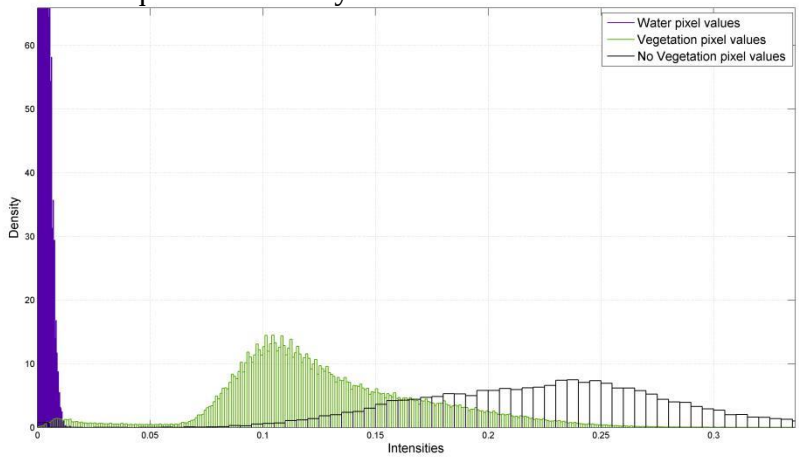


Figure 42 Estimates of Spectral Signatures after the use of ML approximation

An example of the way classification works can be seen by a histogram of values for the intensities of the pixels that are assigned in the three classes. In this histogram we isolated only a single band in 1711nm. It is obvious that the main problem is the mixed pixels between class "vegetation" and "no vegetation". However if we consider e.g. 50 histograms of this kind (one per band), the conditions for correct classification are quite satisfactory.



8. Conclusions

During the preparation and writing of this dissertation the basic concepts and principles of spectroscopy and remote sensing were studied. The state of the art in Hyperspectral imaging was analyzed and a process was developed in order to extract and use information from satellite hyperspectral images.

The result of this study was the MATLAB implementation of four classification algorithms in order to identify areas from satellite images based on their environmental features. We were able to identify areas with water resources, vegetation as well as dry soil or urban/suburban environment. The importance of this work is self-evident as these algorithms could be used to monitor the environmental and climatic changes. The four classification algorithms used were a) Euclidean Minimum Distance, b) Mahalanobis Minimum Distance c) Bayesian Classifier and d) an approximation of the Maximum Likelihood Classifier. The conclusions that can be derived by the application of the algorithms are the following:

- Euclidean Minimum Distance algorithm is the fastest but it is not so accurate. It produces relatively good results and it is able to distinguish areas with clear features (e.g. water resources). It has a bias towards the classes with well-defined training sets when much iteration is used.
- Mahalanobis Minimum Distance algorithm was slightly disappointing. Although the computational workload was increased it its outperformed by the Euclidean Minimum Distance algorithm. This was caused by the fact that the algorithm needs good estimates based on the training samples and the "no-vegetation" samples in our case were limited in number.
- Bayesian algorithm worked very well. It was relatively fair regarding the mixed pixels and the results were very accurate. A small bias towards the class with the smallest number of pixel was shown due to the fact that there is no knowledge of a priori probabilities. However the algorithm was stable and reliable.
- A Maximum Likelihood Approximation algorithm was developed. First a single run of Euclidean Minimum Distance algorithm was executed in order to estimate the a priori probabilities and then these estimates were used to an ML classifier. The results were exceptionally accurate, reliable and unbiased. The algorithm outperforms all the others and it has an additional example. It needs less iteration to achieve better results and therefore it may be faster than the Bayesian and Mahalanobis classifier.

The spectral signatures were also estimated for the defined classes. If ML approximation algorithm is chosen then estimates of the spectral signatures can be seen in Figure 42. The results seem very accurate if we compare it with the results in [51], [54] and [66]. The results are practically identical if we use a normalization

factor. Moreover if we examine the figure from [51] it can be concluded that the vegetated areas have trees, grass or crops with fresh leafs. As far as the spectral signature for dry soil is concerned it is concluded that during the application of the algorithm and the incorporation of mixed pixels the spectral signature tends to a similar shape with the one of the vegetated areas with greater intensity values in wavelengths greater than 1500nm.

Further work could include the use of different images with higher resolution in order to distinguish more classes, application of more advanced classification algorithms, and reduction of data using feature extraction methods. In fact PCA algorithm was developed but no results were extracted due to temporal and spatial limitations.

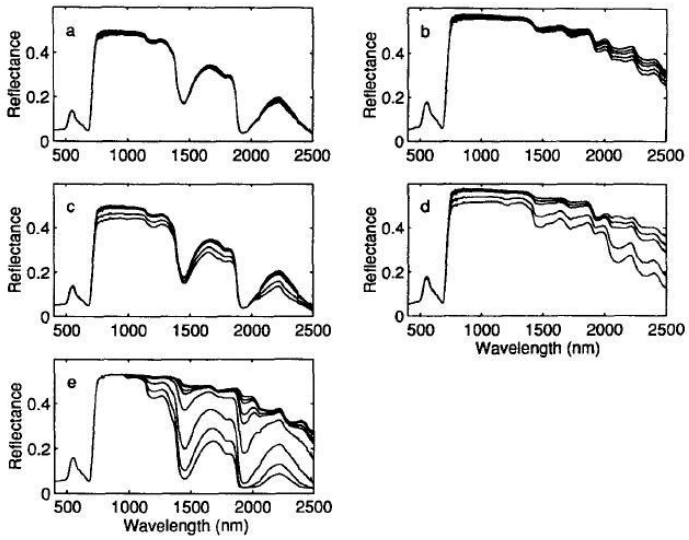


Figure 43: Results from [51] regarding leaf biochemistry

References

[1]: P. Shippert, “Introduction to Hyperspectral Image Analysis”, Online Journal of Space Communication, Earth Science Applications Specialist Research Systems Inc.

[2]: John P. Kerekes and Jerrold E. Baum, “Hyperspectral Imaging System Modeling”, LINCOLN LABORATORY JOURNAL, Vol 14, No 1, 2003

[3]: R.N. Clark, “Manual of Remote Sensing”, John Wiley and Sons Inc. 1999

[4]: R.N. Clark and T.L. Roush, “Reflectance Spectroscopy: Quantitative Analysis Techniques for Remote Sensing Applications”, Journal of Geophysical Research, Vol. 89, No. B7, pp. 6329-6340, 1984

[5]: Pavel Paclik and Robert P.W Dwin, “Dissimilarity based classification of spectra: computational issues”, Real-Time Imaging, Vol. 9, Issue 4, August 2003, Pages 237–244

[6]: Konstantinos Douros, “Analysis of Satellite Hyperspectral Images for the study of quantitative and qualitative physical parameters with the use of GIS”, PhD thesis for Aristotelion University of Thessaloniki

[7]: Athanassios Themelis, “Classification of Urban Surface via Hyperspectral Imaging Systems in Remote Sensing”, Diploma Thesis for National Technical University of Athens

[8]: David Landgrebe, “Signal Theory Methods in Multispectral Remote Sensing”, John Wiley and Sons

[9]: Clark, R.N., T.V.V. King, M. Klejwa, G.A. Swayze, and N. Vergo, 1990, “High Spectral Resolution Reflectance Spectroscopy of Minerals,” Journal of Geophysical Research, Vol. 95

[10]: Vincent, R.K., “Fundamentals of Geological and Environmental Remote Sensing”, Prentice Hall, 1997.

[11]: J. H. Lumme, “Classification of vegetation and soil using imaging spectrometer data”, Institute of Photogrammetry and Remote Sensing, Commission VII.

[12]: Lilesland and Kiefer (1987) “Remote Sensing Image Interpretation. 2nd Edition”, John Wiley,

[13]: Pieter Kempeneers, “Information Extraction from Hyperspectral Images Applied to Vegetation”, PhD Thesis, University of Antwerp, 2007

[14]: R. Congalton, “A review of Assessing the Accuracy of Classifications of Remotely Sensed Data”, Remote Sens. Environ. Vol.37, pp 35-46, 1991

[15]: Boardman, J. W., “Automated spectral unmixing of AVIRIS data using concept geometry concepts”, Proceedings from the Airborne Geosciences Workshop, 1993

[16]: Hord, R. M., “Digital Image Processing of Remotely Sensed Data”, Academic Press, 1982

[17]: C. A. Shah, P. Watanachaturaporn, P. K. Varshney, “Some Recent Results on Hyperspectral Image Classification”, Proceedings of IEEE Workshop on Advances in Techniques for Analysis of Remotely Sensed Data, 2003

[18]: Shailesh Kumar, Joydeep Ghosh, and Melba M. Crawford, “Best-Bases Feature Extraction Algorithms for Classification of Hyperspectral Data”, IEEE Trans. Geoscience and Remote Sensing, Volume: 39, Issue:7, pp 1368 - 1379

[19]: Rutherford V. Platt, “A Comparison of AVIRIS and Synthetic Landsat Data for Land Use Classification at the Urban Fringe”, Interim Report Land-Use Change Project, International Institute for Applied Systems Analysis 2002

[20]: M. Egmont-Petersena, D. de Ridderb, H. Handelsca, “Image processing with neural networks—a review”, Pattern Recognition, Volume 35, Issue 10, October 2002, Pages 2279–2301

[21]: Josef Cihlar, Rasim Latifovic, Jean Beaubien, “A comparison of clustering strategies for unsupervised classification”, Canada Centre for Remote Sensing, Ottawa, 1999

[22]: J. Schwarz, and K. Staenz, “Adaptive Threshold for Spectral Matching of Hyperspectral Data”, Canadian Journal of Remote Sensing (2001), Volume: 27, Issue: 3, Pages: 216-224.

[23]: Chein-I Chang, Hsuan Ren, and Shao-Shan Chiang, “Real-Time Processing Algorithms for Target Detection and Classification in Hyperspectral Imagery”, IEEE Transactions on Geoscience and Remote Sensing, Volume: 39 Issue: 4, pp: 760-768, 2001

[24]: Jason Hamel, “Simulation of Spectral Signatures”, Final Report, Center for Imaging Science Rochester Institute of Technology May 1999

[25]: J. Richards and X. Jia “Remote Sensing Digital Image Analysis: An Introduction”, Springer

[26]: Mark L. Chang, Scott A. Hauck “Adaptive Computing in NASA Multi-Spectral Image Processing”, Military and Aerospace Applications of Programmable Devices and Technologies, 1999.

[27]: Erzsebet Merenyi, “The challenges in Spectral Image Analysis: an introduction and review of ANN Approaches”, University of Arizona, Lunar and Planetary Laboratory

[28]: S. Gürola, H. Öktema, T. Özalpa, B. Karasözen , “STATISTICAL LEARNING AND OPTIMIZATION METHODS FOR IMPROVING THE EFFICIENCY IN LANDSCAPE IMAGE CLUSTERING AND CLASSIFICATION PROBLEMS” ISPRS - WGs of Technical Commission VI

[29]: Rory Hutson, “Remote Sensing and Land Classification”, Remote Sensing Group Plymouth Marine Laboratory, a Tutorial Presentation

[30]: Wilfredo E. Lugo-Beauchamp, “Parallelization of Hyperspectral Imaging Classification and Dimensionality Reduction Algorithms”, Master of Science Thesis in Computer Engineering, University of Puerto Rico 2004.

[31]: Aber, J. D., and Martin, M. E., “High spectral resolution remote sensing of canopy chemistry”, In Summaries of the Fifth JPL Airborne Earth Science Workshop, JPL Publication 95-1, v. 1, pp. 1-4 , 1995

[32]: Patrick J. Rauss, “Classification of Spectral Imagery Using Genetic Programming”, Proceedings of the Genetic and Evolutionary Computation Conference, 2000

[33]: Fabrizio Vagni, “Survey of Hyperspectral and Multispectral Imaging Technologies”, The Research and Technology Organisation (RTO) of NATO, 2007

[34]: Suresh Subramaniana, Nahum Gata, Michael Sheffield a, Jacob Barhenb, Nikzad Toomarian, “Methodology for hyperspectral image classification using novel neural network”, Algorithms for Multispectral and Hyperspectral Imagery III, SPIE Vol. 3071, 1997

[35]: George Lemeshewsky, “Interim Progress Report on the Application of an Independent Components Analysis-based Spectral Unmixing Algorithm to Beowulf Computers”, USGS Open-File Report 03-478, 2004

[36]: Ralph Root, Pablo Zarco-Tejada, Carlos Pinilla, Susan Ustin, Raymond Kokaly, Gerry Anderson, and Steve Hager ,"Identification, Classification, and Mapping of Invasive Leafy Spurge Using Hyperion, AVIRIS, and CASI ", NASA Validation Report.

[37]: Marie-Louise O'Connell, Tom Howley, Alan G. Ryder, Marc N. Leger, Michael G. Madden , "Classification of a target analyte in solid mixtures using principal component analysis, support vector machines and Raman spectroscopy.", 2005, Optical Sensing and Spectroscopy Conference.

[38]: Machelle D. Wilson, Susan L. Ustin and David M. Rocke, "Classification of Contamination in Salt Marsh Plants Using Hyperspectral Reflectance", IEEE Transactions on Geoscience and Remote Sensing, Vol.42 No 5, May 2004

[39]: I. Nedeljkovic, "IMAGE CLASSIFICATION BASED ON FUZZY LOGIC", The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 2004

[40]: D.A. Landgrebe, "Information extraction principles and methods for multispectral and hyperspectral remote sensing", Information Processing for Remote Sensing, pp.3 - 37, 1999.

[41]: F.A. Kruse, A.B. Lefkoff, J.W. Boardman, K.B. Heidenbrecht, A.T.Shapiro, J.P. Barloon, and A.F.H. Goetz, "The Spectral Image Processing System SIPS) – Interactive visualization and analysis of imaging spectrometer data: Remote Sensing Environment, v. 44, pp. 145- 163, 1993

[42]: Donghui Li, Mahmood R. Azimi-Sadjadi, and Marc Robinson, "Comparison of Different Classification Algorithms for Underwater Target Discrimination", IEEE Transactions on Neural Networks, Vol. 15, No. 1, 2004

[43]: S. J. Purkis Z R. Pasterkamp, "Integrating in situ reef-top reflectance spectra with Landsat TM imagery to aid shallow-tropical benthic habitat mapping", Springer Coral Reefs, 2004, Vol. 23, pp. 5–20

[44]: Joseph P. Hoffbeck and David A. Landgrebe, "Classification of High Dimensional Multispectral Image Data", Remote Sensing of Environment, Vol. 57, No. 3, pp 119-126, September 1996

[45]: David Landgrebe , "Some Fundamentals and Methods for Hyperspectral Image Data Analysis", SPIE International Symposium on Biomedical Optics, 1999

[46]: Shefali Aggarwal, "PRINCIPLES OF REMOTE SENSING", Satellite Remote Sensing and GIS Applications in Agricultural Meteorology, Proceedings of Training Workshop, July 2003 in Dehra Dun.

[47]: Shirley Morillo Contreras, "A COMPARISON OF RESOLUTION ENHANCEMENT METHODS AS PREPROCESSING FOR CLASSIFICATION OF HYPERSPECTRAL IMAGES", PhD Thesis, Department of Electrical and Computer Engineering University of Puerto Rico 2004

[48]: Lindsay I Smith, "A tutorial on Principal Components Analysis", www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf , February 26, 2002

[49]: David Landgrebe, "Hyperspectral Image Data Analysis as a High Dimensional Signal Processing Problem", IEEE Signal Processing Magazine, Vol. 19, No. 1 pp. 17-28, January 2002.

[50]: Li, Deren; Shan, Jie; Gong, Jianya (Eds.), "Geospatial Technology for Earth Observation", 2009 Springer.

[51]: S. Jacquemoud, S.L. Ustin, J. Verdebout, G. Schmuck, G. Andreoli, B. Hosgood, "Estimating Leaf Biochemistry Using the PROSPECT Leaf Optical Properties Model", Elsevier Remote Sensing of Environment 1995.

[52]: Noam Levin, "Fundamentals of Remote Sensing", 1st Hydrographic Data Management course, IMO - International Maritime Academy 1999

[53]: Gary Shaw and Dimitris Manolakis, "Signal Processing for Hyperspectral Image Exploitation", IEEE Signal Processing Magazine, January 2002.

[54]: Randall Smith, "Introduction to Remote Sensing of Environment (RSE)", White Paper MicroImages Inc, 2012.

[55]: J. H. Lumme, "Classification of Vegetation and Soil using imaging Spectrometer Data", Proceedings of the XXth ISPRS Congress, 2004

[56]: Li Ma, Melba M. Crawford and Jinwen Tian, "Local Manifold Learning-Based k-Nearest-Neighbor for Hyperspectral Image Classification", IEEE Transactions on Geoscience and Remote Sensing Vol. 48, No 11, 2010

[57]: Stefan A. Robila, Lukasz Maciack, "New Approaches for Feature Extraction in Hyperspectral Imagery", IEEE Long Island Systems, Applications and Technology Conference, LISAT 2006.

[58]: Susan M. Schweizer and José M. F. Moura, "Efficient Detection in Hyperspectral Imagery", IEEE Transactions on Image Processing Vol. 10, No 4, 2001

[59]: Leonidas Fountanas, "Principal Components Based Techniques for Hyperspectral Image Data", Master Of Science in Applied Physics from the Naval Postgraduate School, 2004

[60]: Space Computer Corporation, "An Introduction to Hyperspectral Imaging Technology", Couses, 2007 www.cant.ua.ac.be

[61]: Spectrum Mapping LLC, "CSI Remote Sensing – Hyperspectral Imaging Provides "DNA" for Geospatial Information", White Paper

[62]: Gat, N. "Real-time Multi- and Hyper-spectral Imaging for Remote Sensing and Machine Vision: an Overview", Presented at 1998 ASAE Annual International Mtg., Orlando, Florida, July, 1998.

[63]: P. Shippert,"Why Use Hyperspectral Imagery," PERS, Vol. 70, No. 4, 2004, pp. 377-380.

[64]: De Carvalho, O.A. and Meneses, P.R., "Spectral Correlation Mapper (SCM); An Improvement on the Spectral Angle Mapper (SAM)", 9th JPL Airborne Earth Science Workshop, 00-18, 9 p. (2000).

[65]: Nicholas M. Short, "The Landsat Tutorial Workbook, Basics of Satellite Remote Sensing", NASA Reference Publication, 1982

[66]: M. Govender, K. Chetty and H. Bulcock "A review of hyperspectral remote sensing and its application in vegetation and water resource studies", 2007, www.wrc.org.za/downloads/watersa/2007/Apr%2007/2052.pdf

[67]: Li, D., K. Di, and D. Li, "Land use classification of remote sensing image with GIS data based on spatial data mining techniques", International Archives of Photogrammetry and Remote Sensing, 33(B3):238–245, 2000.

[68]: S. Matteoli, M. Diani and G. Corsini, "A tutorial overview of anomaly detection in hyperspectral images", IEEE Aerospace and Electronic Systems Magazine, Vol. 25, Iss.7, pp. 5 – 28, July 2010

[69]: O'Hara C.G, "Remote sensing and geospatial application for wetland mapping, assessment, and mitigation". In proceeding of ISPRS commission 1 Midterm symposium, Integrated Remote Sensing at the Global, Regional and Local scale. 10-15 November 2002.

[70]: Farid Melgani and Lorenzo Bruzzone, "Classification of Hyperspectral Remote Sensing Images with Support Vector Machines", IEEE Trans. on Geoscience and Remote Sensing, Vol. 42, No. 8, 2004

[71]: Athanassios Papoulis, "Probability, Random Variables and Stochastic Processes 3rd Edition", Mcgraw-Hill College, 1991

[72]: Remote Sensing on line tutorial http://www.fas.org/irp/imint/docs/rst/Intro/Part2_4.html

[73]: Remote Sensing on line tutorial http://www.crisp.nus.edu.sg/~research/tutorial/process.htm

[74]: A.Kitching, C. Edge, "Lasers and surgery", British Journal of Anaesthesia, Vol.8, 2003

[75]: University of Minnesota, "Remote Sensing of Natural Resources and Environment", on line Lectures and Readings http://rsclass.gis.umn.edu/

[76]: Canadian Natural Resources CCRS http://www.nrcan.gc.ca/earth-sciences/ on-line presentations

[77]: ENVI software http://www.exelisvis.com/language/en-us/productsservices/envi.aspx

MATLAB References for code from other sources:

- `illuminant`
- `colorMatchFcn`

The functions were taken from Color & Vision Research Laboratory of UCL. http://cvrl.ioo.ucl.ac.uk. Reader may find the functions in the given link.

Also in order to write the following code, ideas, tips, tricks and code advices were used by the following resources:

- R. Gonzalez and R. Woods, S. Eddins, "Digital Image Processing Using MATLAB", Prentice Hall
- Tutorial on transforming hyperspectral images (http://personalpages.manchester.ac.uk/staff/david.foster/Tutorial_HSI2RGB/Tutorial_HSI2RGB.htm l).
- Matlab Newsgroup (e.g. http://www.mathworks.co.uk/ matlabcentral/newsreader/ view_thread/169432)
- Stackoverflow (e.g. http://stackoverflow.com/questions/8108905/matlab-covariance-matrix-computation-for-different-classes)
- Matlab FileExchange Server and particularly the following submissions:
  - Variational Bayesian Inference for Gaussian Mixture Model by Mo Chen
  - Data matrix whitening by Colorado Reed
  - Image Seqmentation by Abioye Samson
  - Kernel PCA by Ambarish Jash
  - Introduction to Classification by Richard

and of course MATLAB Help by Mathworks.

Appendix: Matlab Scripts and Functions

- **convert_hyper2rgb.m**

This script reads a hyperspectral image and converts it to RGB:

```
% Image Name and folder:
```

Two string variables are defined: folder_name and file_name. Variable folder_name contains the path to the image and files folder and file_name contains the image name without file extension

```
folder_name = 'C:\Users\User\Desktop\hyper_spectral\';
file_name =
'EO1H1890172010184110KC_167_spcmm_atm_polish_smcorr_ortho_scaled';
```

The two variables are merged with strcat:

```
fname = strcat(folder_name, file_name);
```

read_info_from_header is called with fname as input. This function returns the necessary data to call matlab function multibandread. Output: samples is the number of pixels per image line, lines the number of lines, bands is the number of bands, offset is a variable that indicates if a header offset is kept in the image file, ftype is the image file type (bil, bip or bsq) and byte_order shows if the most significant bit is in the start or the end of each image byte.

```
% Read info from header:
[samples, lines, bands, offset, ftype, byte_order] =
read_info_from_header(fname);
```

read_wavelengths_per_band is called with fname as input. It reads the wavelengths for each band from the header file. Wavelengths are in micrometers. Also FWHM for each band is read and also some other variables (dgv and ref_sc_fact) that indicates if the reflectances in the image files are normalized by a factor or not.

```
[wavelengths, fwhm, dgv, ref_sc_fact] = read_wavelengths_per_band(fname);
```

Wavelengths are converted to nanometers with a multiplication by 1000.

```
% wavelengths are(?) in micrometers: do it nano:
wavelengths = 1000*wavelengths;

% In order to go to RGB isolate the bands that are in the visible spectrum
```

A new vector is defined to store the band wavelengths with values less than 830.

```
% define a visible wavelength vector (from the measured wavelengths:
visible_waves = wavelengths(wavelengths<=830); % wavelengths < than 830nm
```

Then all the wavelengths with values greater than 380 nm are kept. This is the visible spectrum.

```
visible_waves = visible_waves(visible_waves>=380); % wavelengths> than 380nm
```

Three matlab cells are defined that specify which subset of the image will be kept.

```
%% Set Image ranges of interest:

%% Define a subset of image Rows-Columns that will be read:
% Line subset:
```

subset_lines is the cell variables defined as specified in multibandread. Parameter 'Range' specifies that all lines from first_line to end_line will be read.

```
first_line = 901;
end_line = 3150;
subset_lines = {'Row','Range',[first_line end_line]};
```

The same is defined for the column subset to be kept.

```
% Sample subset:
first_sample = 1;
end_sample = 1000;
subset_samples =  {'Column','Range',[first_sample end_sample]};
```

For the subset of bands variable parameter 'Direct' is used. This means that multibandread will keep the bands with the indexes contained in the provided vector choose_bands. This vector occurs by keeping all the wavelengths in visible spectrum and by decimating by dec_factor all other wavelengths.

```
% Bands  subset:
dec_factor = 3;
choose_bands = [(1:length(visible_waves))
(length(visible_waves)+2:dec_factor:length(wavelengths))];
subset_bands = {'Band','Direct',choose_bands};
```

We adjust the wavelengths variable to contain only the bands kept.

```
% Adjust wavelengths:
wavelengths = [wavelengths(1:length(visible_waves));
wavelengths(length(visible_waves)+2:dec_factor:end)];
```

multibandread is called with all the data defined previously. The hyperspectral image is kept in Z 3D matrix

```
%% Read Image:
Z = multibandread(strcat(fname,'.',ftype), [lines, samples, bands],
'single', offset, ftype, byte_order, subset_samples, subset_lines,
subset_bands);
```

Then image conversion to RGB is performed:

```
%% Convert to RGB:
```

clorMatchFcn is a function found in Matlab FileExchange from Computer Vision laboratory of UCL. It is a look-up table. As input a specific string is provided with the name of the look-up table to be used. The tables are provided by CIE standards. The output is a set of wavelengths in lambda variable and three valued xFcn, yFcn and zFcn which are the color values of the wavelength in XYZ color format.

```
%% Create/load a vector that provides the relationship between wavelengths
and Colors in XYZ Format
[lambda, xFcn, yFcn, zFcn] = colorMatchFcn('1931_full');
```

Since lambda and wavelengths do not contain the same wavelength values interpolation is performed to estimate X,Y,Z values for wavelengths. Function interp1 is used with interpolation method 'pchip' (piecewise cubic).

```
% From vector lambda with interpolation, find the XYZ colors that
% correspond to the wavelengths in the visible_waves (coming from hdr file):
XYZ = interp1(lambda', [xFcn; yFcn; zFcn]', visible_waves, 'pchip', 0);
```

illuminant is also a function from UCL. It is also a look-up table of the standardized by CIE illuminants. Input is a string that specifies the type of assumed lighting. 'D65' is used that corresponds to medium daylight (logical assumption for satellite image). In the vector energy the illuminant values for wavelength values lambda are kept.

```
%% Radiometric Calibration through illuminants:
% Since the source (sun) does not transmit equally to all bands then we
% will use a standard pattern of flux distribution provided as an
% illuminant:

% Choose illuminant pattern:
illuminant_ptrn = 'D65';
% Create illuminants for lamda wavelengths:
[lambda, energy] = illuminant(illuminant_ptrn);
```

Once again interpolation is performed for the illuminants of visible_waves vector.

```
% Find illuminant values through interpolation for the wavelengths of the
% given band:
energy_new = interp1(lambda, energy, visible_waves, 'pchip',0);

% Apply the illuminant to the matrix, for each band:
```

Illuminants are applied. A new matrix Y is initialized with size equal to the visible spectrum

```
Y = zeros(size(Z,1),size(Z,2),length(visible_waves)); % initialize array
```

For every wavelength of the visible spectrum we find the index in wavelengths vector and we multiply all elements of Z in the specific band with the illuminant. Result is stored in Y

```
for ii = 1:length(visible_waves)
  ind = find( wavelengths == visible_waves(ii));
   Y(:,:,ii) = Z(:,:,ind)*energy_new(ii);
end

% Initialize matrix to store XYZ image:
```

The size of Y (r rows- c columns w-width (bands))

```
[r c w] = size(Y);
```

With reshape function all the pixels are put in a sequential order and matrix Y is converted to 2D

```
Y = reshape(Y, r*c, w);
```

Hyperspectral image is converted to XYZ by matrix multiplication. The intensities for each band are multiplied with corresponding XYZ values (in matrix XYZ) and then summed to give a total resultant. This is the pixel color in XYZ.

```
% Apply the XYZ colors for each band and multiply with reflectance:
Y = (XYZ'*Y')';
```

Matrix Y is reshaped in the image shape and stored in X

```
X = reshape(Y, r,c,3);
```

Values XYZ of MATLAB must be from 0 to 1. The values from the previous multiplication may be greater than 1 because the intensity values may be normalized by a factor. Normalization is performed:

```
% Normalize matrix to fix luminosity:
```

In case there are negative values, those are zeroed:

```
X = max(X,0);
```

max(X(:)) calculates the maximum value of the 3D matrix and the elements are normalized:

```
X = X/max(X(:));
```

Function MATLAB makecform creates a MATLAB struct that can be used to transform an image from XYZ to RGB format. Application of the transform is done by applycform function. sRGB stores the image:

```
%% Convert image from XYZ format to RGB from Matlab functions:
XYZ2sRGB = makecform('xyz2srgb');
sRGB = applycform(X, XYZ2sRGB);

% Show image (in RGB):
imshow(sRGB);
```

Finally the bands of the visible spectrum are also decimated by dec_factor. The result is stored in a file.

```
% Save results to matrices:
Z = Z(:,:,[(1:dec_factor:length(visible_waves))
length(visible_waves)+1:end]);
```

```
wavelengths = wavelengths([(1:dec_factor:length(visible_waves))
length(visible_waves)+1:end]);
save('image_matrices.mat', 'Z', 'wavelengths')
```

MATLAB Functions for Reading the Image:

- **`read_info_from_header:`**

Function reads useful information from header file:

```
function [samples, lines, bands, offset, ftype, byte_order] =
read_info_from_header(fname)
```

strcat(fname,'.hdr') puts extension hdr in a file. fileread puts a text file in a string variable (named text)

```
text = fileread(strcat(fname, '.hdr'));
```

strfind finds the starting index of the first appearance of word ('samples')

```
samp_ind = strfind(text, 'samples');
```

The text segment from variable text from `samp_ind + length('samples') + 3` to `samp_ind +length('samples') + 8` is isolated. The word samples is bypassed and 3 more symbols (' = ')

```
samples = text(samp_ind + length('samples') + 3 : samp_ind
+length('samples') + 8);
```

Since it is not known from the beginning when the sample number ends, the symbols are read one by one and with the use of parameter ('digit') in matlab function isstrprop it is concluded that a symbol is a digit or not. When the loop ends ii variance contains the index of the last digit +1.

```
ii = 1;
while isstrprop(samples(ii), 'digit')
    ii = ii + 1;
end
```

Samples 1 to ii-1 contain the wanted sample number. Function str2double translates the isolated string to number:

```
samples = str2double(samples(1:ii-1));
```

The exactly same procedure is used to find the number of lines:

```
lines_ind = strfind(text, 'lines');
lines = text(lines_ind + length('lines') + 5 : lines_ind + length('lines') +
10);
ii = 1;
while isstrprop(lines(ii), 'digit')
    ii = ii + 1;
end
lines = str2double(lines(1:ii-1));
```

The exactly same procedure is used to find the number of bands:

```
bands_ind = strfind(text, 'bands');
bands = text(bands_ind + length('bands') + 5 : bands_ind + length('bands') +
10);
ii = 1;
while isstrprop(bands(ii), 'digit')
    ii = ii + 1;
end
bands = str2double(bands(1:ii-1));
```

The image file may contain some header bytes. The number of those bytes is stored in the header file in header offset field. This value is extracted and stored to variable offset

```
offset_ind = strfind(text, 'header offset');
offset = text(offset_ind + length('header offset') + 3 : offset_ind +
length('header offset') + 8);
ii = 1;
while isstrprop(offset(ii), 'digit')
    ii = ii + 1;
end
offset = str2double(offset(1:ii-1));
```

Three text samples are identified after the field 'interleave' that contains the file type. Three possible values: BIP, BIL and BSQ. This string is stored in variable ftype:

```
ftype_ind = strfind(text, 'interleave');
ftype = text(ftype_ind + length('interleave') + 3 : ftype_ind +
length('interleave') + 5);
```

The byte order variable specifies the byte ordering (machine format) in which the data is stored. If it is 0 then the byte ordering format is IEEE little endian and the value 'ieee-le' is stored. If it is 1 then the byte ordering format is big endian and the value 'ieee-be' is stored:

```
byte_order_ind = strfind(text, 'byte order');
byte_order = str2double(text(byte_order_ind + length('byte order') + 3 :
byte_order_ind + length('byte order') + 4));
if byte_order == 0
    byte_order = 'ieee-le';
else
    byte_order = 'ieee-be';
end
```

- **read_wavelengths_per_band**

This function extracts the central wavelengths for each band and other information

```
function [wavelengths, fwhm, dgv, ref_sc_fact] =
read_wavelengths_per_band(fname)
```

strcat(fname,'.hdr') adds the extension and fileread stores the header file in a string variable

```
% Input: Filename with no extension
text = fileread(strcat(fname, '.hdr'));
```

Function strfind finds the string ('wavelength = {') contained in text variable.

```
% Find the word 'wavelength = '
start_ind = strfind(text, 'wavelength = {');
% then find }
```

Then the index where character } is found:

```
stop_ind = strfind(text(start_ind + length('wavelength = {') :end), '}');
```

In variable wavelengths all the content between { and } is stored:

```
% Throw away anything not important - keep only numbers:
wavelengths = text(start_ind + length('wavelength = {') + 1 : start_ind +
length('wavelength = {') + stop_ind(1));
```

Matlab function textscan scans the defined text segment and creates a matrix A. In this case the string segment contains numbers (floats '%f' parameter) that are delimited by comma (delimiter parameter is ,). The result in A is the passed to variable wavelengths.

```
% Transform the string to vector using , as delimiter:
```

```
A = textscan(wavelengths, '%f', 'delimiter',','
); wavelengths = A{1};
```

Same procedure is followed for fwhm. FWHM is actually an equivalent of the bandwidth for each image band.

```
% Find the word 'fwhm = '
start_ind = strfind(text, 'fwhm = {');
stop_ind = strfind(text(start_ind + length('fwhm = {') :end), '}');
% Use the same method to construct the fwhm vector:
fwhm = text(start_ind + length('fwhm = {') + 1 : start_ind + length('fwhm
= {') + stop_ind(1));
A = textscan(fwhm, '%f', 'delimiter',',' );
fwhm = A{1};
```

The same procedure is followed for data_gain_values. Data gain values is a field containing numbers that multiply the intensities for each band. It does some kind of scaling if data gain values are different for each band.

```
% Find the word 'data gain values = '
start_ind = strfind(text, 'data gain values = {');
stop_ind = strfind(text(start_ind + length('data gain values = {') :end),
'}');
% Use the same method to construct the dgv vector:
dgv = text(start_ind + length('data gain values = {') + 1 : start_ind +
length('data gain values = {') + stop_ind(1));
A = textscan(dgv, '%f', 'delimiter',',' );
dgv = A{1};
```

Finally the set of words 'reflectance scale factor' is found. This is a scalar value that normalizes the intensities for all the bands.

```
% Find the word ''reflectance scale factor '
start_ind = strfind(text, 'reflectance scale factor = ');
ref_sc_fact = text(start_ind + length('reflectance scale factor = ') :
start_ind + length('reflectance scale factor = ') -1 + 9);
% Use the same method to construct the ref_sc_fac vector:
ii = 1;
while isstrprop(ref_sc_fact(ii), 'digit') || strcmp(ref_sc_fact(ii), '.')
    ii = ii + 1;
end
ref_sc_fact = str2double(ref_sc_fact(1:ii-1));
```

Matlab Script to calculate spectral signatures and covariance matrices from training set:

- **create_spectral_signatures.m**

This script estimates the spectral signatures, the mean intensities for each band and each class. Training sets are defined in order to proceed with estimation. Training sets were chosen by visual examination. One or more areas with e.g. water pixels were selected in rectangular forms.

The first defined class is "water" with pixels from areas with water resources:

```
%% CLASS 1: Water class (train set):
% x-axis coordinates (column)
```

Two matrices Xwater and Ywater are define. Xwater, Ywater are 3x2 matrices. Each line of Xwater and Ywater defines a rectangular. For example a rectangle starting from column 971 to 1000 and from line 743 to 785 is defined first. Inside these rectangular shapes, the training set pixels are contained.

```
Xwater = [971 1000; 695 707; 272 294];
% y-axis coordinates (line)
Ywater = [743 785; 1091 1110; 2984 3030];
% This means that e.g. the rectangular with columns 734 until 785 and lines
```

```
% 971 to 1000 contains pixel from the specific class...
```

Based on the coordinates of the rectangles an Nx2 matrix is defined containing the X and Y coordinated for all the pixels of the training set. This is done with function find_pixel_coords_from_rects

```
% Create a matrix with the coordinates of all the pixels in the
class: water_pixels = find_pixel_coords_from_rects(Xwater, Ywater);
```

Since we now have all the pixels containing in the training set, the spectral signatures can be calculated with the function calculate_mean_spectr_signature with inputs: image Z, X coordinates of the training set, Y coordinates of the training set. In variable water the spectral signature is stored while in water_var the variance per band for the specific class is stored.

```
% Calculate mean reflectance spectrum for the pixels of the class:
[water, water_var] = calculate_mean_spectr_signature(Z, water_pixels(:,1),
water_pixels(:,2));
```

The exact process if followed for class “vegetation”:

```
%% CLASS 2: Vegetation:
% x-axis coordinates
Xgreen = [945 962; 1045 1052; 408 417; 966 979; 415 436; 1064 1077; 555
565;640 678;];
% y-axis coordinates
Ygreen = [379 393; 530 539; 2341 2349; 508 522; 2327 2339; 187 203; 2182
2209; 1751 1761;];

% Create a matrix with the coordinates of all the pixels in the
class: green_pixels = find_pixel_coords_from_rects(Xgreen, Ygreen);

% Calculate mean reflectance spectrum for the pixels of the 1st training set
for the class:
[green, green_var] = calculate_mean_spectr_signature(Z, green_pixels(:,1),
green_pixels(:,2));
```

The exact process if followed for class “no vegetation”:

```
%% CLASS 3: No vegetation

% x-axis coordinates (column)
Xland = [55 62; 665 668;954 956];
% y-axis coordinates (line)
Yland = [3074 3085; 1941 1947;922 928];

% Create a matrix with the coordinates of all the pixels in the
class: land_pixels = find_pixel_coords_from_rects(Xland, Yland);

% Calculate mean reflectance spectrum for the pixels of the 1st training set
for the class:
[land, land_var] = calculate_mean_spectr_signature(Z, land_pixels(:,1),
land_pixels(:,2));
```

In order to have a single variable for the spectral signatures all the results are stored in a 3x(number of bands) matrix where 3 is the number of defined classes.

```
%% Create a single matrix to store the Spectral signatures:
spec_signatures = [water; green; land];
var_signatures = [water_var; green_var; land_var;];
```

A matrix with the classes’ names is created:

```
class_names = [{'water'}; {'green'}; {'land'};];
```

A figure is created to present the estimated spectral signatures for all classes:

```
%% Create Figure:
figure1 = figure;
```

```
% Create axes
axes1 = axes('Parent',figure1,'YGrid','on','XGrid','on');
box(axes1,'on');
hold(axes1,'all');
% Create multiple lines using matrix input to
plot plot1 = plot(wavelengths,[water; green;
land;],'Parent',axes1,'LineWidth',2);
% Create xlabel
xlabel({'Wavelengthts(nm)'},'FontSize',14);
% Create ylabel
ylabel({'Reflectance'},'FontSize',14);
% Create legend

set(plot1(1),'DisplayName','Water');
set(plot1(2),'DisplayName','Vegetation');
set(plot1(3),'DisplayName','No Vegetation');

legend1 = legend(axes1,'show');
set(legend1,...
    'Position',[0.755092592592593 0.743335951819847 0.134722222222222
0.153508771929825]);
```

The sequence of commands for the creation of plot was extracted by the "Generate M-file" option of the MATLAB figure.

- **create_cov_mtx_per_class.m**

This script extracts the covariance matrices for each class. Covariance matrices are used from Mahalanobis and Bayes algorithms. The computation is based on the given training set for each class and a big part of the script is very similar with the create_spectral_signatures script.

```
%% IDENTIFY DIFFERENT CLASSES AND EXTRACT SPECTRAL SEQUENCES
```

All about the rectangles that contain the training set pixels for a class are mentioned in script create_spectral_signatures

```
%% CLASS 1: Water class (train set):
% x-axis coordinates
Xwater = [971 1000; 695 707; 272 294];
% y-axis coordinates
Ywater = [743 785; 1091 1110; 2984 3030];
```

Similarly with create_spectral_signatures the matrix with all the pixels of the class is created:

```
% Create a matrix with the coordinates of all the pixels in the
class: water_pixels = find_pixel_coords_from_rects(Xwater, Ywater);
```

Now the covariance matrix for each class can be calculated from the function calculate_cov_mtx_per_class with input Z (the image), X (x-axis coordinates for the training set, 1st column of water_pixels), Y (y-axis coordinates for the training set, 2nd column of water_pixels) and the spectral signature (mean values per band) for the specific class. In cov_water the covariance matrix is stored.

```
% Calculate covariance matrix for the pixels of the class:
cov_water = calculate_cov_mtx_per_class(Z, water_pixels(:,1),
water_pixels(:,2), spec_signatures(1,:));
```

The same procedure is followed for "vegetation" class:

```
%% CLASS 2: GREEN
% x-axis coordinates
Xgreen = [945 962; 1045 1052; 408 417; 966 979; 415 436; 1064 1077; 555 565;
640 678];
% y-axis coordinates
```

```
Ygreen = [379 393; 530 539; 2341 2349; 508 522; 2327 2339; 187 203; 2182
2209; 1751 1761];

% Create a matrix with the coordinates of all the pixels in the
class: green_pixels = find_pixel_coords_from_rects(Xgreen, Ygreen);

% Calculate covariance matrix for the pixels of the class:
cov_green = calculate_cov_mtx_per_class(Z, green_pixels(:,1),
green_pixels(:,2), spec_signatures(2,:));
```

The same procedure is followed for "no-vegetation" class:

```
%% CLASS 3:NO VEGETATION
% x-axis coordinates
Xland = [55 62; 665 668;954 956];
% y-axis coordinates
Yland = [3074 3085; 1941 1947;922 928];

% Create a matrix with the coordinates of all the pixels in the
class: land_pixels = find_pixel_coords_from_rects(Xland, Yland);

% Calculate mean reflectance spectrum for the pixels of the 1st training set
for the class:
cov_land = calculate_cov_mtx_per_class(Z, land_pixels(:,1),
land_pixels(:,2), spec_signatures(3,:));
```

The number of classes is:

```
%% Create a single matrix to store the Spectral signatures:
num_classes = 3;
```

In order not to have different variables for the covariance matrix of each class a 3D matrix is defined. The first dimension of the new matrix is the index of each class. Therefore cov_matrices has dimensions (num_classes)x(number of bands)x(number of bands). E.g. in cov_matrices(1,:,:) the covariance matrix for "water" class is contained.

```
cov_matrices = zeros(num_classes, size(Z,3), size(Z,3));
cov_matrices(1,:,:) = cov_water;
cov_matrices(2,:,:) = cov_green;
cov_matrices(3,:,:) = cov_land;
```

Matlab Functions to calculate spectral signatures and covariance matrices from training set:

- **find_pixel_coords_from_rects.m**

As mentioned in create_spectral_signatures Xclass and Yclass variables are Nx2 matrices where N is the number of rectangles used as a training set for a class. This function's objective is to create a (number of pixels)x2 matrix with the coordinates of all training set pixels.

```
function pixel_coords = find_pixel_coords_from_rects(Xclass, Yclass)

% This function extracts the coordinates for all the pixels inside a
% rectangular.
% Input contains the coordinates for rectangular. The rectangulars contain
% a set of pixels that belong to a class training set

% Input is a Nx2 Xclass and a Nx2 Yclass matrix where N is the number of
% rectangulars
```

Number of rectangles is equal with the number of Xclass (or Yclass) lines:

```
% Check if the input dimensions are correct
N = size(Xclass,1);
```

A check is performed in order to examine if Xclass and Yclass dimensions are compatible.

```
if size(Yclass,1)~=N || size(Xclass,2)~=2 || size(Yclass,2)~=2
    error('X, Y have not the proper dimensions')
end
```

The initially empty matrix pixel_coords will contain the indexes (coordinates) of the training set pixels.

```
pixel_coords =[];

% For each rectangular defined from Xwater, Ywater do:
```

For every rectangle do:

```
for k = 1 : size(Xclass,1)
    % For all the pixels from column Xwater(k,1) to Xwater(k,2) do:
```

For every pixel of column Xclass(k = current rectangle,1) and till Xclass (k,2) do:

```
    for ii = Xclass(k,1) : Xclass(k,2)
        % For all the pixels from line Ywater(k,1) to Ywater(k,2) do:
```

For every pixel of line Yclass(k = current rectangle,1) and till Yclass (k,2) do:

```
        for jj = Yclass(k,1) : Yclass(k,2)
```

Store the specific pixel's coordinated as a new line in pixel_coords variable:

```
            pixel_coords = [pixel_coords; ii jj];
        end
    end
end
```

- **calculate_mean_spectr_signature.m**

This function calculates the spectral signatures (means per band) for the training sets of each class. Input is: Z(hyperspectral image), X (x-axis coordinates), Y(y-axis coordinates for the pixels).

```
function [spec_signatures, variance_per_band] =
calculate_mean_spectr_signature(Z, X, Y)

% This function calculates the mean spectral signature based on a training
% set provided by the user
% Input:
%   Z = the hyperspectral image
%   X = a Nx2 vector. It gives the x-axis coordinates for the training set
%       N is the number of pixels of image used for the training set
%   Y = the y-axis coordinates
```

The numbers of pixels that take part in the computation of the spectral signature are equal to the number of lines of X (or Y).

```
% number of pixels:
N = size(X,1);
```

Two line vectors are initialized with num. of bands elements.

```
% Initialize matrices to store the variables:
spec_signatures = zeros(1, size(Z,3));
variance_per_band = zeros(1, size(Z,3));
```

For each pixel do:

```
for ii = 1 : N
    % Calculate mean reflectance spectrum for the pixels of the class:
    % Also calculate the variance...
```

The isolated pixel has coordinates Y(ii), X(ii). It is emphasized that X axis are the 2nd dimension (columns) of a matlab matrix. Variable tmp is squeezed from 1x1x(num. of bands) to 1x(num. of bands) . Transpose operator ' transforms tmp in (num. of bands)x1.

```
    tmp = squeeze(Z(Y(ii), X(ii), :)).';
```

The band values for each pixel are summed in vector spec_signatures after they are normalized by the number of pixels N in the training set. The result is the desired mean value:

```
spec_signatures = spec_signatures + tmp/N;
```

Variance is given by the formula E(x^2)-(E(x))^2. First the mean value for the squared intensities for each band is calculated. This is done similarly with before. Every squared value is divided with N and added to variance_per_band. The final result will be E(Z.^2).

```
    variance_per_band = variance_per_band + tmp.^2/N;
end
```

Since variance is given by E(Z^2)-(E(Z))^2 the squared value of the mean (spectral signature) must be subtracted.

```
% To calculate the variance we must subtract the square of mean
value: variance_per_band = variance_per_band - spec_signatures.^2
```

- **calculate_cov_mtx_per_class.m**

This function calculates the covariance matrices for each class for a given training set. Inputs are: Z (the hyperspectral image), X the x-axis coordinates for the pixels of the training set, Y the y-axis coordinates of the pixels and means – the spectral signature of the specific class.

```
function cov_mtx = calculate_cov_mtx_per_class(Z, X, Y, means)

%  This function calculates the covariance matrix per class based on a
training set provided by the user
% Input:
%   Z = the hyperspectral image
%   X = The coordinates in x-axis of the training set
%   Y = The coordinates in y-axis of the training set
```

The output matrix is initialized with zeros and dimensions (number of bands) x (number of bands)

```
cov_mtx = zeros(size(Z,3), size(Z,3));
```

A counter is also initialized at zero. This counter will count the processed pixels.

```
counter = 0;
```

For each pixel of the training set do:

```
for ii = 1 : length(X)
```

Pixel Z(Y(ii), X(ii),:) is isolated. This matrix has 1x1x(number of bands) dimensions. In order to be used in matrix multiplications the matrix is squeezed to (number of bands)x1. Estimation of the covariance matrix for a vector $\mathbf{z}$ with mean $\mathbf{m}$ is performed by the following formula:

$$E\left((\mathbf{z}-\mathbf{m})(\mathbf{z}-\mathbf{m})^T\right)=\sum_{\text{for all pixels}}\frac{(\mathbf{z}-\mathbf{m})(\mathbf{z}-\mathbf{m})^T}{N}$$

Since the vector is (num of bands)x1 then after the multiplication the resulted matrix has dimensions (num of bands)x(num of bands). Since means is a line vector we use 'operator to transpose. The multiplication result is stored in matrix tmp $(\mathbf{z}-\bar{\mathbf{m}})(\mathbf{z}-\bar{\mathbf{m}})^T$ for each pixel.

```
        tmp = (squeeze(Z(Y(ii), X(ii), :))-means')*(squeeze(Z(Y(ii),
X(ii), :))-means')';
```

tmp is added in cov_mtx which is initially zero. The results from all pixels are cumulated to cov_mtx. `cov_mtx = cov_mtx + tmp;`

For each pixel processed, the counter is increased by one.

```
        counter = counter + 1;
end
```

According to the formula the covariance matrix is estimated as a mean value so it must be divided by the number of pixels

```
cov_mtx = cov_mtx/counter;
```

Matlab Functions used before classification:

- **find_im_limits.m**

As seen the processed MATLAB image is diagonal. Therefore a big part of the loaded image does not contain useful information and it is a background with zeros for all bands. It is not desired to process and classify those pixels and therefore the pixel indexes for each line that contains useful information are extracted.

```
function lim_x = find_im_limits(Z)

% Since the image is diagonal, we find the actual pixels of the image in
% order to ignore the black background during classification:
```

In variable Y we store the number of lines of matrix Z (lines of the image). A (number of lines)×2 matrix is initialized. lim_x will contain the indexes of useful information pixels per line.

```
Y = size(Z,1);
lim_x = zeros(Y,2);
% Sum the absolute values of the intensities for all bands...
% If the sum is non-zero, then the pixel contains info
```

The absolute value of the intensities for all bands is taken and then the sum for each pixel is calculated. Matrix Z that occurs has dimensions (num. lines)×(num. columns). If a pixel is in the background then the sum is also zero (as a sum of zeros).

```
Z = sum(abs(Z),3);
```

For each line of the image, function find is used to find the pixel positions with non-zero sum of absolute values. The result is contained in vector ind. The first element indicates the position of the first information pixel and the last the position of the last information pixel of the line:

```
for ii = 1 : Y
    ind = find(Z(ii, :) >0);
```

First it is checked if no non-zero pixels are found:

```
    if ~isempty(ind)
        lim_x(ii,1) = ind(1);
        lim_x(ii,2) = ind(end);
    else
```

If vector ind is empty then we put the values 1 and 0 for the starting and ending position. Since 0<1 when the specific values are used, no pixel of the image will be selected.

```
        lim_x(ii,1) = 1;
        lim_x(ii,2) = 0;
    end
end
```

- **scaling_hyp_image.m**

Scaling is performed by function scaling_hyp_image. Inputs are: the spectral signatures for all classes, variances per band for all classes and string “method” that defines the method used for scaling.

In matlab, varargin is a special structure used for variable length of input arguments. When it is used it means that one or more input arguments may or may not exist. In this case it means that the input arguments must be at least 3.

```
function [scaled_signatures, scaled_vars, scaled_vect] =
scaling_hyp_image(spec_signatures, var_signatures, method, varargin)

% A function that scales the bands of a hyperspectral image, so that all
% the bands have the same influence for minimum distance algorithms:

%% Normalize by setting to 1 the maximum value for each band:
```

Two methods are defined. If we compare the string method with 'max2one' and the result is true:

```
if strcmp(method, 'max2one')
```

Method max2one finds the maximum value of intensity in every band for each class. With the use of max MATLAB function a vector scaled_vect (1x (num. bands)) is defined that will be used for normalization

```
    scaled_vect = max(spec_signatures);
```

Normalization is performed with scaled_vect for all classes (a matrix with 3x(num bands) dimensions is created with the use of matlab function repmat)

```
    scaled_signatures = spec_signatures./repmat(scaled_vect,
size(spec_signatures,1), 1);
```

Variable scaled_signatures stores the spectral signature for a scaled image with vector scaled_vect. In order to calculate the variances for the scaled image, normalization must also be performed but in this case with square scaled_vect:

```
    scaled_vars = var_signatures./repmat((scaled_vect).^2,
size(spec_signatures,1),1);

elseif strcmp(method,'var')
```

The second method of scaling is based in variance ('var'). In this case scaling for each band is done with the maximum variance of each band. Usually a multiplier is also used in order to create a normalization factor of the following form: $\sqrt{a\sigma^2}$ where σ^2 is the variance and a the multiplier. Thus for 'var' method a fourth input is necessary. In this point matlab reads the fourth input from varargin and it stores it in variable alpha.

```
    alpha = varargin{1};
```

If there is no fourth input, MATLAB produces an error.

```
    if isempty(varargin)
        error('If using var method for scale, you must also give an alpha
factor')
    end
```

In this case the scaling vector per band occurs from the maximum value of variance of all classes per band multiplied by factor alpha. Then the squared root of the vector is taken $\sqrt{a\sigma^2}$

```
   scaled_vect = sqrt(max(alpha*var_signatures));
```

Computation of scaled signatures and scaled variances is performed with the same procedure as in 'max2one' method.

```
    scaled_signatures = spec_signatures./repmat(scaled_vect,
size(spec_signatures,1),1);
    scaled_vars = var_signatures./repmat((scaled_vect).^2,
size(spec_signatures,1),1);
end
```

Matlab Functions used for classification:

- **euc_min_distance_class.m**

This function is an implementation of the Euclidean Minimum Distance classification algorithm. Inputs:

Z: the hyperspectral image

class_centers: the spectral signatures without scaling, (num classes)x(num bands)

var_classes: the variances per band for each class without scaling

num_iter: the number of iterative runs that will be executed

lim_x : the matrix that contains the information pixels for each image line with dimensions (number lines)x2

null_class_factor: This factor defines the rule according to which a given pixel is classified to a null class or not. For this algorithm is a scalar value. If the distance from all means for each band is less than this value multiplied by standard deviation then this pixel is assigned to a class or else it is classified as null. If no null class is wanted then null_class_factor = inf

assign_all_pix: It is a flag. It can be either 1 or 0.

varargin: It is used if it is necessary to load more inputs in the function. If scaling is performed then two extra inputs must be assigned. Else no extra input is necessary.

```
function [Zmap Zclass] = euc_min_distance_class(Z, class_centers,
var_classes, num_iter, lim_x, null_class_factor, assign_all_pix, varargin)

% Number of classes:
```

The number of classes is the first dimension of class_centers:

```
num_classes = size(class_centers, 1);
% Check if scaled data are used or not:
```

If varargin is not an empty cell (~ =not isempty) then do:

```
if ~isempty(varargin)
```

In this case scaling is used. As 8th input the scaling vector scaled_vect is used. This is the 1st element of varargin

```
    % If scaled:
    scaled_vect = varargin{1};
```

The 9th input is the method used for scaled_vect (max2one or var)

```
    method = varargin{2};
```

If scaling is used then class_centers and var_classes are recalculated for every class. First two matrices in the same dimensions (scaled_vect and scaled_vars) are initialized as zero matrices.

```
    % Recalculate class centers and class variances:
    scaled_centers = zeros(size(class_centers));
    scaled_vars = zeros(size(var_classes));
```

For every class do:

```
    for kk = 1: num_classes % Do this for each
        class % Rescale center:
```

The spectral signature is recalculated by dividing class_centers with scaled_vect element-wise

```
        scaled_centers(kk,:) = class_centers(kk,:)./scaled_vect; %
        Rescale variance (must divide with squared scaled_vect)
```

Similarly var_classes are divided by squared scaled_vect:

```
        scaled_vars(kk,:) = var_classes(kk,:)./(scaled_vect.^2);
    end

else
```

If varargin is empty then no scaling is used and thus scaling_vect is a vector of ones. Variables scaled_centers and scaled_vars are the same with class_centers and var_classes.

```
    scaled_vect = ones(1, size(Z,3));
    % Since no scaling is used, scaled centers and vars are equal with the
    % original:
    scaled_centers = class_centers;
    scaled_vars = var_classes;
end

% Calculate pixel number, by finding how many pixels there are in each row
% and then sum the useful pixels:
```

With the use of lim_x the total number of useful pixels is calculated. E.g. for the ii-th line the useful pixels are the difference of the 2nd column of the ii-th line from the 1st column of the same line. Vector `((lim_x(:,2) - lim_x(:,1)+1))` is the information pixels per line. With sum function the total number of useful pixels is calculated:

```
pixel_number = sum((lim_x(:,2) - lim_x(:,1)+1));
```

Then pixels are isolated from the 3D matrix Z and a new 2D matrix is created with line number equal with the number of useful pixels and columns the number of bands. It is initialized with zeros.

```
% Put all the useful pixels in a sequential row
Znew = zeros(pixel_number, size(Z,3));
```

A counter is defined. Initialized as 1:

```
counter = 1;
```

For every image line do:

```
for k = 1 : size(Z,1)
```

Pixels of the k-th line with information pixels are isolated with the use of lim_x matrix. Counter in each step of the loop defines the index in Znew where the useful pixels are placed. For k=1 pixels are placed from 1 to lim_x(1,2)-lim_x(1,1) +1 .

```
    Znew(counter : counter + lim_x(k,2) - lim_x(k,1),:) = Z(k,
lim_x(k,1):lim_x(k,2),:);
```

Then the counter increases so that the pixels of the next iteration are placed in the correct place. For example for the 2nd iteration the index for the first new pixel is `lim_x(1,2) - lim_x(1,1)+2`

```
    counter = counter + lim_x(k,2) - lim_x(k,1) + 1;
end
```

Variable changed_pix is used when more than a single run is performed, num_iter >1. In case there is no change in the classification of a pixel from iteration to another, there is no need to continue the iterative loop. For the first run, changed_pix is set to one in order to perform the first (and last) execution of the algorithm.

```
changed_pix =1;
```

A matrix with size (number of pixel)x1 is initialized to store the classification results.

```
% Initialize Zclass to classify pixels
Zclass = zeros(size(Znew,1),1);
```

Also a matrix with size (number of lines)x(number of columns) is initialized. The classification results will also be stored in it. This matrix is actually a visual map for the pixels in image dimensions. The values of Zclass and Zmap are 1 for “water”, 2 for “vegetation”, 3 for “no vegetation” and 0 for null (if used).

```
% Initialize Zmap to store the classes:
Zmap = zeros(size(Z,1), size(Z,2));
```

A counter that counts the number of iterations performed is initialized as one.

```
counter = 1;
```

As long as pixels are changing classes and counter is less than num_iter do:

```
while (changed_pix ~= 0) && counter<=num_iter
```

In case this is not the first iteration the previous classification result is stored in Zclass_old

```
    if counter>1
        Zclass_old = Zclass;
    end
```

For each pixel do:

```
    for ii = 1 : pixel_number
```

Euclidean distance for pixel with values Znew(ii,:) is calculated after it is properly scaled with a division with scaled_vect. Euclidian distance is the sum of squared differences of the scaled pixel from the scaled spectral signatures for each class. In order to do that in a single step, Znew(ii)./scaled_vect is repeated 3 time (in 3 rows, function repmat) and is subtracted from scaled_center variable.

```
        % Calculate distance from centers:
        distance = sum(abs(repmat(Znew(ii,:)./scaled_vect, num_classes, 1) -
scaled_centers).^2,2);
```

By calling the min function the class is found by the second output ind which class minimizes the Euclidean distance (1, 2 or 3)

```
        % Find the minimum:
        [min_value,ind] = min(distance);
```

A check is performed if the condition for null class classification is valid.

```
        % Check if a pixel is in null class or not
```

Differences from Minimum distance are recalculated:

```
        check_dist = (Znew(ii,:)./scaled_vect - scaled_centers(ind,:));
```

If the absolute value for each element of check_dist is smaller than null_class_factor*sqrt(scaled_vars(ind,:)) then the pixel is not in the null class. Each element is compared with the threshold. If the sum is equal to zero it means that for all bands the condition is satisfied. If it is greater than zero then the pixel is a null class pixel.

```
        if sum(abs(check_dist) > null_class_factor*sqrt(scaled_vars(ind,:)))
> 0
```

Pixel is classified as null:

```
            Zclass(ii) = 0;
```

In case the last iteration is performed (counter == num_iter) and the flag assign_all_pix is 1 then null class is not used and all pixels are assigned. The final result will not contain null pixels (except of the background).

```
            if assign_all_pix == 1 && (counter == num_iter)% if we want to
assign all the pixels in the final step....
                    Zclass(ii) = ind;
            end
        else
```

If all band distances are smaller than null_class_factor*sqrt(scaled_vars(ind,:))) then pixels are classified normally:

```
            Zclass(ii) = ind;
        end

    end
```

If we are in iteration greater than the first:

```
    if counter>1
```

The number of changed classification results is calculated by the difference Zclass_old - Zclass and check if the result is difference than zero. The summation of the check results is the number of changed pixels.

```
        changed_pix = sum((Zclass_old - Zclass ~=0));
```

The result is shown in the command window:

```
        display(changed_pix)
    end
```

The results of Zclass are placed in Zmap matrix in order to reshape the map as an image.
A new counter is defined, counter2.

```
    % Create class map:
    counter2 = 1;
```

For each image line till size(Z,1) do:

```
    for k = 1 : size(Z,1)
```

Pixels are positioned in Zmap in the place they were taken from Z. It is an inverse procedure of the one creating Zclass vector.

```
        Zmap(k, lim_x(k,1):lim_x(k,2)) = Zclass(counter2 : counter2 +
lim_x(k,2) - lim_x(k,1),:);
```

Variable counter2 increases so that each pixel of Zclass is placed in the correct line and column.

```
        counter2 = counter2 + lim_x(k,2) - lim_x(k,1) + 1;
    end
```

Classification is completed and counter is increased by one.

```
    counter = counter + 1;
```

Assuming that classification is done correctly the spectral signatures are recalculated.

```
    % Recalculate spectral signatures
    class_vars = zeros(num_classes, size(Z,3));
```

For each class:

```
      for k = 1 : num_classes
```

Find the pixels assigned to class k:

```
        [ind1 ind2] = find(Zmap == k);
        % Calculate mean reflectance spectrum for the pixels of the 1st
training set for the class:
```

Call calculate_mean_spectr_signature to re-estimate spectral signature where (ind2, ind1) are (x,y):

```
        [tmp,tmp2] = calculate_mean_spectr_signature(Z, ind2, ind1);
```

The results are placed in the proper rows of class_centers and
class_vars. `class_centers(k,:) = tmp;`

```
        class_vars(k,:) = tmp2;
    end
```

If varargin is not empty (scaling) then the new scaling vector scaled_vect must be calculated:

```
    if ~isempty(varargin)
```

Υπολογίζω τις τιμές για μετά το scaling:

```
        [scaled_centers, scaled_vars, scaled_vect] =
scaling_hyp_image(class_centers, class_vars, method);
    end
```

If counter is less than num_iter then classification is performed again with the new spectral signatures.

```
end
```

- **mahal_min_distance_class.m**

This function is an implementation of the Mahalanobis minimum distance algorithm. Only the differences regarding the implementation of the Euclidean distance algorithm are presented.

1st Difference: Instead of the variances as input the covariance matrices for each class is used (cov_matrices)

```
function Zmap = mahal_min_distance_class(Z, class_centers, cov_matrices,
num_iter, lim_x, null_class_factor, assign_all_pix, varargin)

% Number of classes:
num_classes = size(class_centers, 1);
```

```
% Check if scaled data are used or not:
if ~isempty(varargin)
    % If scaled:
    scaled_vect = varargin{1};
    method = varargin{2};

    % Recalculate class centers and class variances:
    scaled_centers = zeros(size(class_centers));
    scaled_vars = zeros(size(class_centers));

    for kk = 1: num_classes % Do this for each
        class % Rescale center:
        scaled_centers(kk,:) = class_centers(kk,:)./scaled_vect;
        % Rescale covariance matrix ( = A*C*A where A
= diag(1./scaled_vect))
```

2nd Difference: Instead of scaling the variances, the covariance matrix is getting scaled. According to theory of covariance matrices this is done as follows: Vector 1./scaled_vect is becoming diagonal (function diag) and multiplies covariance matrix (tmp) from left and right.

```
        tmp = squeeze(cov_matrices(kk,:,:));
        tmp = diag(1./scaled_vect)*tmp*diag(1./scaled_vect);
        cov_matrices(kk,:,:) = tmp;
```

3rd Difference: Scaled variances can be found in the diagonal of the covariance matrix.

```
% Scaled variance is the diagonal of the covariance
        matrix: scaled_vars(kk, :) = diag(tmp);
    end
else
    scaled_vect = ones(1, size(Z,3));
    % Since no scaling is used:
    scaled_centers = class_centers;
end

% Calculate pixel number, by finding how many pixels there are in each row
% and then sum the useful pixels:
pixel_number = sum((lim_x(:,2) - lim_x(:,1)+1));

% Put all the useful pixels in a row
Znew = zeros(pixel_number, size(Z,3));
counter = 1;
for k = 1 : size(Z,1)
    Znew(counter : counter + lim_x(k,2) - lim_x(k,1),:) = Z(k,
lim_x(k,1):lim_x(k,2),:);
    counter = counter + lim_x(k,2) - lim_x(k,1) + 1;
end

changed_pix =1;

% Initialize Zclass to classify pixels
Zclass = zeros(size(Znew,1),1);
% Initialize Zmap to store the classes:
Zmap = zeros(size(Z,1), size(Z,2));

counter = 1;
while (changed_pix ~= 0) &&
    counter<=num_iter if counter>1
        Zclass_old = Zclass;
    end
    for ii = 1 : pixel_number
        % Calculate  Mahalanobis  distance
        distance = zeros(num_classes,1);
```

4th Difference: Since each class has its own covariance matrix a loop is created to calculate Mahalanobis distance.

```
        for jj = 1 : num_classes
            % Invert the Covariance matrix:
```

5th Difference: Each covariance matrix is loaded in variable tmp. Since tmp is 1x(num_bands)x(num_bands) it is squeezed to (num_bands)x(num_bands):

```
            tmp = (squeeze(cov_matrices(jj,:,:)))^-1;
            % Calculate distance from centers:
```

6th Difference: Mahalanobis Distance. This is: $(\mathbf{z}-\mathbf{m})^T \mathbf{C}^{-1} (\mathbf{z}-\mathbf{m})$

For each pixel the mean value is subtracted and the covariance matrix is multiplied from right and left (in transpose). But first scaling is performed:

```
            distance(jj) =   (Znew(ii,:)./scaled_vect -
scaled_centers(jj,:))*tmp*(Znew(ii,:)./scaled_vect - scaled_centers(jj,:))';
        end
        % Find the minimum:
        [~,ind] = min(distance);

        end

        % Check if a pixel is in null class or not
```

The minimum Mahalanobis distance is recalculated:

```
        tmp = (squeeze(cov_matrices(ind,:,:)))^-1;
        check_dist = (Znew(ii,:)./scaled_vect -
scaled_centers(ind,:))*tmp*(Znew(ii,:)./scaled_vect -
scaled_centers(ind,:))';
```

7th Difference: In Euclidean distance null_class_factor is a multiplier of the variance for each band. In this case null_class_factor is the exact threshold defining the Mahalanobis Distance that classifies a pixel to null. The way it is defined can be found in the proper chapter of the dissertation.

```
        if sum(sqrt(abs(check_dist)) > null_class_factor) > 0
            Zclass(ii) = 0;
            % Define intermediate classes?
            if assign_all_pix == 1 && (counter == num_iter)% if we want to
assign all the pixels in the final step....
                Zclass(ii) = ind;
            end
        else
            Zclass(ii) = ind;
        end

        if mod(ii, 1000)==0
            display(ii)
        end
    end
    if counter>1
        changed_pix = sum((Zclass_old - Zclass
        ~=0)); display(changed_pix)
    end

    % Create class map:
    counter2 = 1;
    for k = 1 : size(Z,1)
        Zmap(k, lim_x(k,1):lim_x(k,2)) = Zclass(counter2 : counter2 +
lim_x(k,2) - lim_x(k,1),:);
        counter2 = counter2 + lim_x(k,2) - lim_x(k,1) + 1;
    end
    counter = counter + 1;

    class_vars = zeros(num_classes, size(Z,3));
    % Recalculate spectral signatures
    for k = 1 : num_classes
        [ind1 ind2] = find(Zmap == k);
        % Calculate mean reflectance spectrum for the pixels of the 1st
training set for the class:
        [tmp,~] = calculate_mean_spectr_signature(Z, ind2,
        ind1); class_centers(k,:) = tmp;
```

8th Difference: After re-estimating the spectral signature based on the classification results, a new covariance matrix is also estimated from calculate_cov_mtx_per_class function.

```
        tmp = calculate_cov_mtx_per_class(Z, ind2, ind1, tmp);
        cov_matrices(k,:,:) = tmp;
        class_vars(k,:) = diag(tmp);
    end
    if ~isempty(varargin)
        [scaled_centers, scaled_vars, scaled_vect] =
scaling_hyp_image(class_centers, class_vars, method);
```

9th Difference: Scaling is also applied to the covariance matrix estimates.

```
        for kk = 1 : num_classes
            tmp = squeeze(cov_matrices(kk,:,:));
            tmp = diag(1./scaled_vect)*tmp*diag(1./scaled_vect);
            cov_matrices(kk,:,:) = tmp;
        end
    end
end
```

- **bayes_class.m**

The function implements the Bayes classification algorithm. Inputs:

Z: the hyperspectral image

class_centers: the spectral signatures without scaling, (num classes)x(num bands)

var_classes: the variances per band for each class without scaling

num_iter: the number of iterative runs that will be executed

lim_x : the matrix that contains the information pixels for each image line with dimensions (number lines of image)x2

null_class_prob: The value provided is not a distance value or a multiplier of the variance but a probability. It defines that if the probability a pixel to be a part of a class is less than this threshold (assuming the Gaussian distribution and estimates are accurate) then the pixel is classified as null. In this case a value in the area of 0.1-0.15 was used.

assign_all_pix: It is a flag. It can be either 1 or 0.

No varargin is used since there is not a reason to use scaling.

```
function Zmap = bayes_class(Z, class_centers, cov_matrices,
num_iter, lim_x, null_class_prob, assign_all_pix)
```

The number of classes is the first dimension of class_centers:

```
num_classes = size(class_centers, 1);
```

With the use of lim_x the total number of useful pixels is calculated. E.g. for the ii-th line the useful pixels are the difference of the 2nd column of the ii-th line from the 1st column of the same line. Vector `((lim_x(:,2) - lim_x(:,1)+1))` is the information pixels per line. With sum function the total number of useful pixels is calculated:

```
pixel_number = sum((lim_x(:,2) - lim_x(:,1)+1));
```

Then pixels are isolated from the 3D matrix Z and a new 2D matrix is created with line number equal with the number of useful pixels and columns the number of bands. It is initialized with zeros.

```
% Put all the useful pixels in a row
Znew = zeros(pixel_number, size(Z,3));
```

A counter is defined. Initialized as 1:

```
counter = 1;
```

For every image line do:

```
for k = 1 : size(Z,1)
```

Pixels of the k-th line with information pixels are isolated with the use of lim_x matrix. Counter in each step of the loop defines the index in Znew where the useful pixels are placed. For k=1 pixels are placed from 1 to lim_x(1,2)-lim_x(1,1) +1 .

```
    Znew(counter : counter + lim_x(k,2) - lim_x(k,1),:) = Z(k,
lim_x(k,1):lim_x(k,2),:);
```

Then the counter increases so that the pixels of the next iteration are placed in the correct place. For example for the 2nd iteration the index for the first new pixel is `lim_x(1,2) - lim_x(1,1)+2`

```
    counter = counter + lim_x(k,2) - lim_x(k,1) + 1;
end
```

Variable changed_pix is used when more than a single run is performed, num_iter >1. In case there is no change in the classification of a pixel from iteration to another, there is no need to continue the iterative loop. For the first run, changed_pix is set to one in order to perform the first (and last) execution of the algorithm.

```
changed_pix =1;
```

A matrix with size (number of pixel)x1 is initialized to store the classification results.

```
% Initialize Zclass to classify pixels
Zclass = zeros(size(Znew,1),1);
```

Also a matrix with size (number of lines)x(number of columns) is initialized. The classification results will also be stored in it. This matrix is actually a visual map for the pixels in image dimensions. The values of Zclass and Zmap are 1 for “water”, 2 for “vegetation”, 3 for “no vegetation” and 0 for null (if used).

```
% Initialize Zmap to store the classes:
Zmap = zeros(size(Z,1), size(Z,2));
```

A counter that counts the number of iterations performed is initialized as one.

```
counter = 1;
```

As long as pixels are changing classes and counter is less than num_iter do:

```
while (changed_pix ~= 0) && counter<=num_iter
```

In case this is not the first iteration the previous classification result is stored in Zclass_old

```
    if counter>1
        Zclass_old = Zclass;
    End
```

For each pixel do:

```
    for ii = 1 : pixel_number
```

Vector probability with number of elements equal to the number of classes is initialized. In each element the probability the pixel (Z(y,x,:)) belongs to a class is stored. For example 1st element will store the probability the pixel belongs to the 1st class.

```
        probability = zeros(num_classes,1);
```

For each class the probability is calculated:

```
        for jj = 1 : num_classes
```

Each covariance matrix is loaded in variable tmp. Since tmp is 1x(num_bands)x(num_bands) it is squeezed to (num_bands)x(num_bands):

```
            tmp = (squeeze(cov_matrices(jj,:,:)));
            % Calculate distance from centers:
```

Assuming that pixels are multivariate Gaussian random variables, the probability the pixel is member of a class is calculated by the formula: $p = \frac{1}{2\pi\sqrt{|\mathbf{C}|}} e^{-\frac{(\mathbf{z}-\mathbf{m})^T \mathbf{C}^{-1} (\mathbf{z}-\mathbf{m})}{2}}$ with $\mathbf{m}$ the spectral signature for each class.

```
            probability(jj) = (1/2/pi)*sqrt(1/det(tmp))*exp(-
(Znew(ii,:) - class_centers(jj,:))*(tmp^-1)*(Znew(ii,:) -
class_centers(jj,:))'/2);
        end
```

The class that maximizes the probability is found using max function from the second output argument ind.

```
        % Find the maximum:
        [max_value,ind] = max(probability);
```

If this probability is less than a threshold probability value the pixel is assigned to null class.

```
        if max_value < null_class_prob
            Zclass(ii) = 0;
```

In case the last iteration is performed `(counter == num_iter)` and the flag assign_all_pix is 1 then null class is not used and all pixels are assigned. The final result will not contain null pixels (except of the background).

```
            if assign_all_pix == 1 && (counter == num_iter)% if
we want to assign all the pixels in the final step....
                Zclass(ii) = ind;
            end
        else
```

If the probability is less than null_class_prob then pixels are classified normally:

```
            Zclass(ii) = ind;
        end
    end
```

If we are in iteration greater than the first:

```
    if counter>1
```

The number of changed classification results is calculated by the difference Zclass_old - Zclass and check if the result is difference than zero. The summation of the check results is the number of changed pixels.

```
        changed_pix = sum((Zclass_old - Zclass
        ~=0)); display(changed_pix)
    end
```

The results of Zclass are placed in Zmap matrix in order to reshape the map as an image.
A new counter is defined, counter2.

```
    % Create class
    map: counter2 = 1;
```

For each image line till size(Z,1) do:

```
    for k = 1 : size(Z,1)
```

Pixels are positioned in Zmap in the place they were taken from Z. It is an inverse procedure of the one creating Zclass vector.

```
        Zmap(k, lim_x(k,1):lim_x(k,2)) = Zclass(counter2 :
counter2 + lim_x(k,2) - lim_x(k,1),:);
```

Variable counter2 increases so that each pixel of Zclass is placed in the correct line and column.

```
        counter2 = counter2 + lim_x(k,2) - lim_x(k,1) + 1;
    end
```

Classification is completed and counter is increased by one.

```
    counter = counter + 1;
```

Assuming that classification is done correctly the spectral signatures are recalculated. For each class:

```
    % Recalculate spectral signatures
    for k = 1 : num_classes
```

Find the pixels assigned to class k:

```
        [ind1 ind2] = find(Zmap == k);
```

Call calculate_mean_spectr_signature to re-estimate spectral signature where (ind2, ind1) are (x,y)

```
% Calculate mean reflectance spectrum for the pixels of the 1st
training set for the class:
        [tmp,~] = calculate_mean_spectr_signature(Z, ind2, ind1);
```

The result is placed in the proper row of class_centers

```
        class_centers(k,:) = tmp;
```

Covariance matrices are also re-estimated by calculate_cov_mtx_per_class function:

```
        tmp = calculate_cov_mtx_per_class(Z, ind2, ind1,
        tmp); cov_matrices(k,:,:) = tmp;
    end

end
```

- **mle_class.m**

In order to classify using the ML approximation algorithm the following script is used:

The Euclidean minimum distance algorithm is executed for a single run with the use of a null class: (the value of null_class_factor is an example).

```
num_iter = 1;
null_class_factor = 3.2;
assign_all_pix = 0;
Zmap = euc_min_distance_class(Z, spec_signatures,var_signatures, num_iter,
lim_x, null_class_factor,assign_all_pix,scaled_vect, method);
```

An estimate of the a priori probabilities is extracted by dividing the total number of pixels classified in each class with the total number of pixels classified:

```
apriori = zeros(1,3);
total_classified = sum(Zmap~=0); apriori(1)
= sum(Zmap==1)/total_classified;
```

```
apriori(2) = sum(Zmap==2)/total_classified;
apriori(3) = sum(Zmap==3)/total_classified;
```

The final result is given to function mle_class (for num_iter, assign_all_pix and null_class_prob selected by the user – the values here are an example):

```
num_iter = 2;
assign_all_pix = 0;
null_class_prob = 0.15;
Zmap3 = mle_class(Z, spec_signatures, cov_matrices, num_iter, apriori,
lim_x, null_class_prob,assign_all_pix);
```

Function mle_class is exactly the same with bayes_class with two simple changes. A priori probabilities are given as input and the probability that is used for classification is given by the following command:

```
probability(jj) = apriori(jj)*(1/2/pi)*sqrt(1/det(tmp))*exp(-(Znew(ii,:) -
class_centers(jj,:))*(tmp^-1)*(Znew(ii,:) - class_centers(jj,:))'/2);
```

Main Matlab script to produce the simulation results:

```
convert_hyper2rgb
close all
create_spectral_signatures

close all
create_cov_mtx_per_class

% Find image limits:
lim_x = find_im_limits(Z);

% Scaling picture
method = 'max2one';
[scaled_signatures, scaled_vars, scaled_vect] =
scaling_hyp_image(spec_signatures, var_signatures, method);

%% Minimum Euclidean distance for scaled image:
num_iter = 4;
null_class_factor = 3.2;
assign_all_pix = 1;
Zmap = euc_min_distance_class(Z, spec_signatures,var_signatures, num_iter,
lim_x, null_class_factor,assign_all_pix,scaled_vect, method);

colorset = zeros(num_classes+1, 3);
colorset(1,:) = [0 0 0];
colorset(2,:) = [16 51 184]/256;
colorset(3,:) = [6 109 40]/256;
colorset(4,:) = [152 103 7]/256;

Zmap_tmp = ind2rgb(Zmap+1, colorset);
image(Zmap_tmp);

%% Mahalanobis distance for scaled image:
num_iter = 4;
null_class_factor = 9;
assign_all_pix = 1;
Zmap2 = mahal_min_distance_class(Z, spec_signatures, cov_matrices, num_iter,
lim_x, null_class_factor, assign_all_pix, scaled_vect, method);

colorset = zeros(num_classes+1, 3);
colorset(1,:) = [0 0 0];
colorset(2,:) = [16 51 184]/256;
colorset(3,:) = [6 109 40]/256;
colorset(4,:) = [152 103 7]/256;

Zmap_tmp2 = ind2rgb(Zmap2+1, colorset);
```

```
figure
image(Zmap_tmp2);

%% Bayesian classifier:
num_iter = 4;
assign_all_pix = 1;
null_class_prob = 0.15;
Zmap3 = bayes_class(Z, spec_signatures, cov_matrices, num_iter, lim_x,
null_class_prob,assign_all_pix);

colorset = zeros(num_classes+1, 3);
colorset(1,:) = [0 0 0];
colorset(2,:) = [16 51 184]/256;
colorset(3,:) = [6 109 40]/256;
colorset(4,:) = [152 103 7]/256;

Zmap_tmp3 = ind2rgb(Zmap3+1, colorset);
figure
image(Zmap_tmp3);

%% MLE classifier (Mdistance and Bayes)
num_iter = 1;
null_class_factor = 3.2;
assign_all_pix = 0;
Zmap = euc_min_distance_class(Z, spec_signatures,var_signatures, num_iter,
lim_x, null_class_factor,assign_all_pix,scaled_vect, method);

apriori = zeros(1,3);
total_classified = sum(Zmap~=0); apriori(1)
= sum(Zmap==1)/total_classified; apriori(2)
= sum(Zmap==2)/total_classified; apriori(3)
= sum(Zmap==3)/total_classified;

num_iter = 2;
assign_all_pix =1;
null_class_prob = 0.15;
Zmap4 = mle_class(Z, spec_signatures, cov_matrices, num_iter, apriori,
lim_x, null_class_prob,assign_all_pix);

colorset = zeros(num_classes+1, 3);
colorset(1,:) = [0 0 0];
colorset(2,:) = [16 51 184]/256;
colorset(3,:) = [6 109 40]/256;
colorset(4,:) = [152 103 7]/256;

Zmap_tm4 = ind2rgb(Zmap4+1, colorset);
figure
image(Zmap_tmp4);
```

Other MATLAB functions created in the project:

- **pca_decomposition.m**

Function to find the PCA decomposition matrix:

```
function [pca_trans, D] = pca_decomposition(cov_mtx)
% Implements Principal Component Analysis - For data reduction

% Do the eigenvalue decomposition
[V, D] = eig(cov_mtx);

% Sort the eigenvalues:
% min to max:
[D, ind] = sort(diag(D));
% max to min:
```

```
D = D(end:-1:1); ind
= ind(end:-1:1);

% Do D matrix
again: D = diag(D);

% Define the pca transformation matrix:
pca_trans = V(:,ind);
```

- **perform_pca_trans.m**

Function to perform PCA decomposition:

```
function ZZ = perform_pca_trans(Z, lim_x, cov_mtx, min_thres)

%% Perform the pca transformation and only keep the significant channels

 % Calculate the PCA transformation matrix:
[pca_trans, D] = pca_decomposition(cov_mtx);

%% Isolate the usefull pixels:
pixel_number = sum((lim_x(:,2) - lim_x(:,1)+1));
% Put all the useful pixels in a row
Znew = zeros(pixel_number, size(Z,3));
counter = 1;
for k = 1 : size(Z,1)
    Znew(counter : counter + lim_x(k,2) - lim_x(k,1),:) = Z(k,
lim_x(k,1):lim_x(k,2),:);
    counter = counter + lim_x(k,2) - lim_x(k,1) + 1;
end


%% Perform the PCA transformation for all pixels:
ZZ = zeros(size(Znew));
% Calculate the mean (for each band for all classes)
for k = 1 : pixel_number
    ZZ(k,:) = pca_trans*Znew(k,:)';
end


%% Keep only the significant channels:
sig_channels = find(diag(D)>D(1,1)/min_thres);

ZZ = ZZ(:,1:length(sig_channels));
```