

The Remote Sensing and GIS Software Library.

Pete Bunting and Daniel Clewley

Documentation and Examples of using RSGISLib.

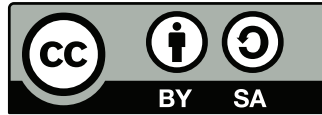
September 28, 2013

Aberystwyth University



Copyright © Pete Bunting and Daniel Clewley 2013.

This work is licensed under the Creative Commons Attribution-ShareAlike 3.0 Unported License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-sa/3.0/>.



Acknowledgements

We would like to thank the developers of the software libraries which RSGISLib depends on. Landcare Research, New Zealand, are thanked for their support for Pete Bunting during his period of employment (2011–2012) in New Zealand during which time RSGISLib was further developed. Mahta Moghaddam (USC) is thanked for supporting Dan Clewley (2012 –) and the Natural Environment Research Council (NERC) for providing funding for his PhD (2009 – 2012). John Armston and James Shepherd are thanked for providing ideas, improvements and bugs fixes. Additionally, all the students and collaborators who have used, tested and reported bugs in RSGISLib are also thanked.

Authors

Peter Bunting

Dr Pete Bunting joined the Institute of Geography and Earth Sciences (IGES), Aberystwyth University, in September 2004 for his Ph.D. where upon completion in the summer of 2007 he received a lectureship in remote sensing and GIS. Prior to joining the department, Peter received a BEng(Hons) in software engineering from the department of Computer Science at Aberystwyth University. Pete also spent a year working for Landcare Research in New Zealand before rejoining IGES in 2012 as a senior lecturer in remote sensing.

Contact Details

EEmail: pfb@aber.ac.uk

Senior Lecturer in Remote Sensing
Institute of Geography and Earth Sciences
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
United Kingdom

Daniel Clewley

Dr Dan Clewley joined IGES in 2006 undertaking an MSc in Remote Sensing and GIS, following his MSc Dan undertook a Ph.D. entitled Retrieval of Forest Biomass and Structure from Radar Data using Backscatter Modelling and Inversion under the supervision of Prof. Lucas and Dr. Bunting. Prior to joining the department Dan completed his BSc(Hons) in Physics within Aberystwyth University. Dan is currently a post-doc researcher at the University of Southern California.

Contact Details

Email: clewley@usc.edu

Postdoctoral Research Associate
Microwave Systems, Sensors, and Imaging Lab (MiXIL)
Ming Hsieh Department of Electrical Engineering
The University of Southern California
Los Angeles
USA

Table of Contents

1	Introduction	1
1.1	Background	1
1.2	Using RSGISLib	1
1.3	The RSGISLib XML Interface	2
1.3.1	XML Basics	2
1.3.2	Escape Characters	3
1.3.3	Commenting	4
1.3.4	RSGISLib XML	4
1.4	Python Bindings	5
1.5	Basic UNIX	6
1.5.1	Editing a text file	7
1.6	Using Batch Queues on a HPC	7
2	Installing RSGISLib	9
2.1	Getting the RSGISLib Source Code	9
2.2	Compiling RSGISLib	10
2.3	Pre-requisites	11
2.3.1	Boost	12

2.3.2	HDF5	12
2.3.3	FFTW	12
2.3.4	Xerces-C	12
2.3.5	MuParser	13
2.3.6	GSL	13
2.3.7	GEOS	13
2.3.8	Proj.4	13
2.3.9	GDAL	13
2.3.10	KEALib	14
2.3.11	GMP	14
2.3.12	MPFR	15
2.3.13	CGAL	15
2.4	Compiling on Windows	15
3	Examples – Image Processing	17
3.1	Basic Landsat Imagery Pre-Processing	17
3.1.1	Convert to Radiance	19
3.1.2	Convert to TOA	21
3.1.3	Re-projecting to OSGB	23
3.1.4	Calculating Statistics and Image Pyramids	24
3.1.5	Calculate an NDVI	24
3.1.6	Expanding the Processing to Multiple Scenes	25
3.2	Linear Spectral Unmixing	30
3.2.1	Defining End Members	31
3.2.2	Unmixing the Scene	32

4	Examples – Image Utilities	35
4.1	Creating a new Image	35
4.1.1	Copying an existing image header	35
4.2	Editing Header information	36
4.2.1	Set the projection	36
4.2.2	Over write spatial header	36
4.2.3	Remove Spatial Header	36
4.3	Stacking Image Bands	36
4.4	Sub-setting Images	37
4.4.1	Subset to Image	37
4.4.2	Subset to Vector	38
4.4.3	Subset to Polygons	38
4.5	Generate Image Tiles	39
4.5.1	Square tiles	39
4.6	Mosaic	39
4.7	Sub-Sample the Image file	40
5	Examples – Vectors	42
5.1	Zonal Statistics	42
5.1.1	Pixel-weighted mean	46
5.1.2	pixelmeanLSSVar	46
5.1.3	Statistics for points	47
5.2	Image Footprints	48
5.3	Vector Maths	48
6	Examples - Image Segmentation	50

6.1	Iterative Elimination Algorithm	50
6.1.1	XML Code	51
7	Examples – Raster GIS	55
7.1	Populating Clumps with Statistics	55
7.1.1	Basic Statistics	55
7.1.2	Calculating Indices	56
7.1.3	Percentiles: Median	58
7.1.4	Location	58
7.1.5	Shape Parameters	59
7.1.6	Relative Border to a Class	60
7.1.7	Populating with an Existing Classification	61
7.2	Classification	61
7.2.1	Rule Based	61
7.2.2	Updating an Existing Classification	62
7.3	Change Detection	62
7.4	Other Utilities	63
7.4.1	Export Columns to Raster Bands	63
7.4.2	Calculate Statistics	64
8	Examples – Image Registration	65
8.1	Tie Point Generation	65
8.2	Image Warping	68
9	Examples – Radar (SAR)	70
9.1	Biomass Estimation	70

10 Examples – Other Utilities	74
10.1 Running Command Line Tools from XML	74
10.1.1 Creating a Directory	74
10.1.2 Deleting Files	74

List of Figures

3.1	Example of a landsat scene which has been unmixed using linear spectral unmixing.	34
-----	---	----

List of Tables

1.1	Keyboard shortcuts for the ne editor.	7
2.1	The pre-requisite software libraries of RSGISLib.	11
3.1	Landsat ETM+ (7) gains and offsets for converting DN's to radiance	19
3.2	Solar irradiance for the Landsat ETM+ bands.	19
5.1	Input file format for polygonImageFootprints	48
9.1	Coefficients for the semi-empirical method of biomass estimation (Saatchi et al., 2007)	73

Chapter 1

Introduction

1.1 Background

The remote sensing and GIS software library (RSGISLib) was developed at Aberystwyth University by Pete Bunting and Daniel Clewley. Development started in April 2008 and has been actively maintained and added to ever since. For more information see <http://www.rsgislib.org>.

1.2 Using RSGISLib

RSGISLib has two interfaces a command line interface, where parameters are provided by an XML file, and python bindings. Additionally a python script is provided with RSGISLib, `rsgiscmd` which generates and executes a temporary XML file. The provides access to most algorithms using standard option flags e.g.,

```
rsgiscmd --algor imagecalc --option imagemaths \  
  --image inImage.kea --output outImage.kea \  
  --format KEA --datatype Float32 \  
  --expression '10*log10(b1)'
```

To see all algorithms, and available options type:

```
rsgiscmd --help all
```

Warning, this prints a lot of output so you may wish to pipe to `less`.

1.3 The RSGISLib XML Interface

The main commands used in the XML interface are:

rsgisexe - the main command to execute scripts

rsgislibxmllist - a command to list all the available commands within the library
(there are over 300!!)

rsgislibcmdxml.py - a command to allow script templates to be populated with
file paths and names.

rsgislibvarsxml.py - a command to input variable values into a template script.

1.3.1 XML Basics

RSGISLib is parameterised through the use of an XML script. XML stands for Extensible Markup Language.

Extensible - XML is extensible. It lets you define your own tags, the order in which they occur, and how they should be processed or displayed. Another way to think about extensibility is to consider that XML allows all of us to extend our notion of what a document is: it can be a file that lives on a file server, or it can be a transient piece of data that flows between two computer systems.

Markup - The most recognizable feature of XML is its tags, or elements (to be more accurate).

Language - XML is a language that's very similar to HTML. It's much more flexible than HTML because it allows you to create your own custom tags. However, it's important to realize that XML is not just a language. XML

is a meta-language: a language that allows us to create or define other languages. For example, with XML we can create other languages, such as RSS, MathML (a mathematical markup language).

```
<parent_element>
  <some_information>
  </some_information>
  <some_information name="some data" value="some other data" />
</parent_element>
```

XML is made up of opening and closing elements, where the hierarchy of the elements provides meaning and structure to the information stored. Therefore, every element has an opening and closing element. This can be defined in two ways; firstly with two tags, where the opening tag is just enclosed with angled brackets (`< tag >`) and the closing tag contains a backslash and angled brackets `< /tag >`. Using this method further tags for data can be stored between the two tags, providing structure as shown above. The second method uses just a single tag with an ending backslash (`< tag/ >`). This second method is used when no data or further tags are to be defined below current element.

```
<element></element>
<element/>
```

1.3.2 Escape Characters

As with all computing languages there are certain characters which have specific meanings and therefore an escape character needs to be used if these characters are required within the input.

& - `&`;

' - `'`;

” - `"`;

< - `<`;

> - `>`;

```
= - &#61;
<element attribute="&apos;hello&apos;" />
  <element>
    1 is &lt; 100
  </element>
<element attribute="&quot;world&quot;" />
```

1.3.3 Commenting

To add comments to XML code and temporarily comment out parts of your XML script you need to use the XML commenting syntax as show below.

```
<!-- Some useful comment -->
<parent_element>
  <some_information>
  </some_information>
  <!-- This is some really useful information in this comment -->
  <some_information name="some data" value="some other data" />
</parent_element>
```

All parts of the document between the opening and closing comment tags will be ignored by the parser.

1.3.4 RSGISLib XML

For parameterisation of the rsgisexe application you will need to create an XML file in the correct format, which the RSGISLib executable understands, while adhering to the rules of XML outlined above. The basis for the RSGISLib XML is to provide a list of commands. Therefore, the XML has the following structure:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Wed Nov 28 15:53:41 2012.
6     Copyright (c) 2012 **Organisation**. All rights reserved.
7 -->
8
```



```
9 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
10
11   <!-- ENTER YOUR XML HERE -->
12   <rsgis:command algor="name" option="algor_option" attr1="foo"
13                                     attr2="bar">
14     <rsgis:data attribute="blob" />
15   </rsgis:command>
16   <rsgis:command algor="algor_name" option="algorithm_option"
17                                     attr="data"/>
18
19 </rsgis:commands>
```

Where all the input parameters are defined using element attributes and each algorithm and option have their own set of attributes to be specified. Within the XML file imported into rsgisexe multiple command elements can be specified and they will all be executed in the order specified in the XML file. Therefore, a sequence of events can be specified and executed without any further interaction.

1.4 Python Bindings

For an increasing number of commands python bindings are available in addition to the XML interface. Incorporating commands directly in python (rather than using python to produce and execute XML files, allows more advanced processing chains to be built. You can use the build in python help to show help for the python bindings. To see available libraries, from an interactive python shell type.

```
1 import rsgislib
2 help(rsgislib)
```

To get help for the imageutils library you can use:

```
1 from rsgislib import imageutils
2 help(imageutils)
```

1.5 Basic UNIX

When interacting with a UNIX terminal you will need to so using a UNIX command line console. The basic UNIX commands for navigating the file system are shown below.

```
# List the current directory
ls

# Enter the directory called 'directory_name'
cd directory_name

# Go down a directory
cd ..

# Display contents of a file
cat file.txt

# Display contents of a file and stop at end of each page
cat file.txt | more

# Display header information for an image
gdalinfo image.kea | more

# Rename file1.txt to file2.txt
mv file1.txt file2.txt

# Move file.txt to within directory output
mv file.txt output/file.txt

# Copy the file file2.txt to file2.txt
cp file1.txt file2.txt

# Copy a directory files1 to files2
cp -R ./files1 ./files2

# Delete a file
rm file.txt

# Delete a directory
```

```
rm -Rf ./files

# List commands previously executed
history

# Edit a text file
ne name_of_text_file.txt
```

1.5.1 Editing a text file

All the scripts you will use to interact with the machine are text files (even if they have the extension `.xml` or `.sh`) to edit them we recommend you use the ‘ne’ editor (‘the nice editor’). This editor uses commonly used desktop keyboard shortcuts for saving files and quitting the application. Table 1.1 lists the common shortcuts while pressing escape will show a menu from which you can select the option you require.

Table 1.1: Keyboard shortcuts for the ne editor.

Function	Shortcut
Saving	ctrl-s
Quit	ctrl-q
Find	ctrl-f
Jump to Line	ctrl-j

1.6 Using Batch Queues on a HPC

For processing large amounts of data, RSGISLib can be run on a High Performance Computer (HPC), such as HPC Wales (<http://www.hpcwales.co.uk>). HPCs use batch priority queue to manage the jobs submitted by multiple users. The job of the queue manages the jobs, which are being submitted to the computer system to make sure that as much of the system as possible is being utilised. Therefore, if you submit a job which is small (i.e., does not use much runtime, memory or a large number of processes) it is likely to run much sooner than a job which requires

a large amount of resources as it will take longer for the scheduler to find space on the system for the large job.

There are a number of command line tools associated with queue facility, including:

```
1 # Submit a job
2 bsub < jobfile.lsf
3
4 # List your jobs submitted
5 bjobs
6
7 # List all jobs in queue
8 bjobs -u all
9
10 # Info on previously run jobs
11 bhist -l <job id>
12
13 # Remove a job from the queue
14 bdel <job id>
```

To submit a job you need to create a job file script which provides the scheduler information such as run time, user account etc such that it can appropriate allocate the job. A template is shown below.

```
1 #!/bin/bash --login
2 #BSUB -J JOB_NAME
3 #BSUB -o JOB_CONSOLE_OUTPUT_FILE.out
4 #BSUB -e JOB_CONSOLE_ERROR_FILE.err
5 #BSUB -W HH:MM (RUN TIME)
6 #BSUB -P sam0004
7 #BSUB -n 1 (NUMBER OF PROCESSING CORES)
8 #BSUB -R "span[ptile=1]"
9
10 # ENTER THE COMMAND WHICH NEEDS TO EXECUTED
11 rsgisexe -x processSomeData.xml
```

Chapter 2

Installing RSGISLib

The notes below hopefully provide some useful details on the process for installing RSGISLib. These notes are intended for people compiling the software on a UNIX platform such as Mac OSX, Linux or Solaris (these are the platforms on which the software has been tested). See the end of the chapter for Windows info, Section 2.4.

To compile the software (and the pre-requisites) you will need a C++ compiler, we use the GNU GCC (<http://gcc.gnu.org>) compilers but the software has also been tested and compiles without a problem using the SunPro compiler on Solaris and the Intel x86 compilers.

You will also need to have mercurial (<http://mercurial.selenic.com>) installed to download the latest version of the RSGISLib source code and cmake <http://www.cmake.org> to configure the source code before compilation.

2.1 Getting the RSGISLib Source Code

The RSGISLib source code is hosted within a Mercurial repository on bitbucket (<https://bitbucket.org/petebunting/rsgislib>). To clone the source code into a folder (rsgislib) run the following command

```
hg clone https://bitbucket.org/petebunting/rsgislib rsgislib
```

2.2 Compiling RSGISLib

If the libraries are all installed in the default location (`/usr/local`) then the following commands are all that is require to build RSGISLib:

```
cmake -D CMAKE_INSTALL_PREFIX=/usr/local .
```

```
make
```

```
make install
```

If libraries are not installed within `/usr/local` then the path needs to be specified using the variables available through the CMake scripts and are listed below.

```
cmake -D CMAKE_INSTALL_PREFIX=/usr/local \  
-D BOOST_INCLUDE_DIR=/usr/local \  
-D BOOST_LIB_PATH=/usr/local \  
-D GDAL_INCLUDE_DIR=/usr/local \  
-D GDAL_LIB_PATH=/usr/local \  
-D HDF5_INCLUDE_DIR=/usr/local \  
-D HDF5_LIB_PATH=/usr/local \  
-D XERCESC_INCLUDE_DIR=/usr/local \  
-D XERCESC_LIB_PATH=/usr/local \  
-D GSL_INCLUDE_DIR=/usr/local \  
-D GSL_LIB_PATH=/usr/local \  
-D FFTW_INCLUDE_DIR=/usr/local \  
-D FFTW_LIB_PATH=/usr/local \  
-D GEOS_INCLUDE_DIR=/usr/local \  
-D GEOS_LIB_PATH=/usr/local \  
-D MUPARSER_INCLUDE_DIR=/usr/local \  
-D MUPARSER_LIB_PATH=/usr/local \  
-D CGAL_INCLUDE_DIR=/usr/local \  
-D CGAL_LIB_PATH=/usr/local \  
-D GMP_INCLUDE_DIR=/usr/local \  
-D GMP_LIB_PATH=/usr/local \  
-D MPFR_INCLUDE_DIR=/usr/local \  
-D MPFR_LIB_PATH=/usr/local \  
-D KEA_INCLUDE_DIR=/usr/local \  
-D KEA_LIB_PATH=/usr/local \  
-D CMAKE_VERBOSE_MAKEFILE=ON \  
.
```

```
make
```

```
make install
```

On some systems you may need to run

```
ldconfig
```

this will set up the dynamic linking paths. Run this if your system returns an error message that the libraries cannot be found when you run `rsgisexe`.

2.3 Pre-requisites

The RSGISLib software library has a number of software prerequisites, which are required to build the software (Table 2.1).

Table 2.1: The pre-requisite software libraries of RSGISLib.

Pre-requisite	License	Website	Input / Output	Usage	Min. Version
GDAL/OGR	MIT-X	http://www.gdal.org		Reading and writing raster and vector datasets.	1.6.X
HDF5	BSD-style	http://www.hdfgroup.org		Reading and writing HDF5 files.	1.8.X
KEALib	MIT-X	http://www.kealib.org		Accessing extra features of the KEA format which are available through GDAL.	1.3.0
Xerces-C	Apache 2.0	http://xerces.apache.org/xerces-c		Parsing XML files.	3.1.1
Mathematical Operations					
GNU Scientific Library (GSL)	GPL 3	http://www.gnu.org/software/gsl		Vector and Matrix operations and other common mathematical tools.	1.13
FFTW	GPL 2	http://www.fftw.org		Fast Fourier transformation.	3.3.X
MuParser	MIT-X	http://muparser.beltoforion.de		Parsing and evaluating mathematical expressions.	2.2.2
Geometry					
GEOS	LGPL 2.1	http://trac.osgeo.org/geos		Spatial geometry operations (e.g., intersect, buffer).	3.X
CGAL	GPL 3	http://www.cgal.org		Triangulations and interpolation.	4.X
Proj.4	MIT-X	https://trac.osgeo.org/proj		Presentation and transformation of projections.	4.8.0
Other					
Boost	Boost	http://www.boost.org		Various utilities (e.g., text processing and save casting) and data structures (e.g., graphs).	1.48

The following sections provide build instructions for the libraries in an appropriate order. The paths and variables in the examples below are for use within the Envmaster (<https://bitbucket.org/chchrsc/envmaster>) system. For a full system installation instructions look at <http://www.massey.landcareresearch.co.nz/client/osgeo/buildinstructions/>

2.3.1 Boost

```
./bootstrap.sh --prefix=/share/osgeo/fw/boost/1.51 \  
--with-libraries=all \  
--with-python=$PYTHON_ROOT/bin/python \  
--with-python-root=$PYTHON_ROOT  
  
./b2  
./bjam install  
mkdir /share/osgeo/fw/boost/1.51/bin  
cp bjam /share/osgeo/fw/boost/1.51/bin
```

2.3.2 HDF5

```
./configure --prefix=/share/osgeo/fw/hdf5/1.8.9 \  
--with-zlib=$ZLIB_ROOT \  
--enable-cxx \  
--enable-shared \  
-with-pic  
  
make  
make install
```

2.3.3 FFTW

```
./configure --prefix=/share/osgeo/fw/fftw/3.3.2.2 \  
--enable-shared \  
--disable-static CPPFLAGS=-fPIC  
  
make  
make check  
make install
```

2.3.4 Xerces-C

```
./configure --prefix=/share/osgeo/fw/xerces-c/3.1.1
```



```
make
make install
```

2.3.5 MuParser

```
./configure --prefix=/share/osgeo/fw/muparser/2.2.2
make
make install
```

2.3.6 GSL

```
./configure --prefix=/share/osgeo/fw/gsl/1.15

make
make install
```

2.3.7 GEOS

```
./configure --prefix=/share/osgeo/fw/geos/3.2.3

make
make install
```

2.3.8 Proj.4

```
./configure --prefix=/share/osgeo/fw/proj4/4.8.0

make
make install
```

2.3.9 GDAL

```
./configure --prefix=/share/osgeo/fw/gdal/1.10.0 \
--with-hdf5=$HDF5_ROOT \
--with-libz=$ZLIB_ROOT \
```

```

--with-xerces=$XERCES_C_ROOT \
--with-geos=$GEOS_BIN_PATH/geos-config \
--with-static-proj4=$PROJ4_ROOT \
--with-python \
--enable-shared=yes --enable-static=no

make
make install

```

2.3.10 KEALib

```

cmake -D CMAKE_INSTALL_PREFIX=/share/osgeo/fw/libkea/1.30 \
-D CMAKE_VERBOSE_MAKEFILE=ON \
-D HDF5_INCLUDE_DIR=$HDF5_INCLUDE_PATH \
-D HDF5_LIB_PATH=$HDF5_LIB_PATH \
-D GDAL_INCLUDE_DIR=$GDAL_INCLUDE_PATH \
-D GDAL_LIB_PATH=$GDAL_LIB_PATH \
.

make
make install

```

You should also set the ‘GDAL_DRIVER_PATH’ variable to the installation path of libkea (i.e., ‘/share/osgeo/fw/libkea/1.30/gdalplugins’). This can be done by adding the following line to your .bash_profile:

```
export GDAL_DRIVER_PATH=/share/osgeo/fw/libkea/1.30/gdalplugins
```

By defining the variable GDAL will be able to read and write KEA files. To test this run the following command ‘gdal_translate -long-usage’ and check that the KEA format is within the list.

2.3.11 GMP

```

./configure --prefix=/share/osgeo/fw/gmp/5.0.5 \
--enable-cxx \
--enable-fft

```

```
make
make install
```

2.3.12 MPFR

```
./configure --prefix=/share/osgeo/fw/mpfr/3.1.1 \
--with-gmp=$GMP_ROOT
```

```
make
make install
```

2.3.13 CGAL

```
cmake -D CMAKE_INSTALL_PREFIX=/share/osgeo/fw/cgal/4.1 \
-D MPFR_INCLUDE_DIR=$MPFR_INCLUDE_PATH \
-D MPFR_LIBRARIES=$MPFR_LIB_PATH/libmpfr.dylib \
-D GMP_INCLUDE_DIR=$GMP_INCLUDE_PATH \
-D GMP_LIBRARIES=$GMP_LIB_PATH/libgmp.dylib \
-D GMP_LIBRARIES_DIR=$GMP_LIB_PATH \
-D GMPXX_INCLUDE_DIR=$GMP_INCLUDE_PATH \
-D GMPXX_LIBRARIES=$GMP_LIB_PATH/libgmpxx.dylib \
-D ZLIB_LIBRARY=$ZLIB_LIB_PATH/libz.1.2.7.dylib \
-D ZLIB_INCLUDE_DIR=$ZLIB_INCLUDE_PATH \
-D CMAKE_VERBOSE_MAKEFILE=ON \
.
```

```
make
make install
```

2.4 Compiling on Windows

There is nothing that should stop you from compiling RSGISLib on Windows but we have not tested it so there would probably be some issues which would need working through to make it happen. This is something we are keen to see happen but we do not have any experience in development under Windows or using Visual

Studio. If you interested we would happily work with anyone wanting to tackle this issue.

The other options are to install using Cygwin (<http://www.cygwin.com>) or to install Linux through virtualisation. VirtualBox (<https://www.virtualbox.org>) is a free and open source virtualisation package with wide support for multiple platforms, including Windows. The UNIX installation instructions should be followed once you have one of these setup.

Chapter 3

Examples – Image Processing

3.1 Basic Landsat Imagery Pre-Processing

The first step when downloading optical imagery is to pre-process it such that it is geographically registered and spectrally it is a measure of reflectance.

You have been provided with four Landsat 7 ETM+ scenes covering parts of Wales. Your task is to:

1. Convert the measured spectral values to at sensor radiance.
2. Convert the measured at sensor radiance to top of atmosphere (TOA) reflectance.
3. Where possible pre-calculate image pyramids and statics for visualisation.
4. Stack the individual Landsat ETM+ bands to create a single image.
5. Re-project the scene to the Ordnance Survey National Grid from the Universal Transverse Mercator (UTM) it is provided in.
6. Calculate the Normalised Difference Vegetation Index (NDVI)

Ideally you would correct the data to surface reflectance but to do so requires the use of an atmospheric model. Atmospheric models require parameterisation

so to simplify the process you will just correct the data to top of atmosphere reflectance.

Top of atmosphere (TOA) reflectance is the ratio of the incoming light, from the Sun (i.e., source), and the light reflected and measured at the sensor. This assumes there is no atmospheric interference or bi-directional reflectance (BRDF) effects within the scene. These are clearly false assumptions but it does provide a simple and fast method of correcting to a measured unit which is corrected for the variation in the solar irradiance, due to season and angle.

To correct the imagery to TOA the first step is to convert the 8 bit (0 – 255) digital number (DN) pixels values to floating point radiance values (i.e., the amount of energy measured at the sensor).

Side Question

What is radiance and what is its unit?^{ab}

^aRadiance is defined as “The power passing through a unit area in a unit solid angle about the normal to the area (per unit spectral interval)”

^bwatts per steradian per square nano metre $W \cdot sr^{-1} \cdot m^{-3}$ or $W \cdot sr^{-1} \cdot nm^{-1}$ for a given wavelength.

To convert DN’s to radiance you need to use Equation 3.1 and the gains and offsets for each landsat 7 band, provided in Table 3.1. The gains and offsets are also available with the MTL text file associated within each Landsat scene downloaded from the USGS.

$$L = \left(\frac{LMAX - LMIN}{QCAL_{max} - QCAL_{min}} \right) (DN - QCAL_{min}) + LMIN \quad (3.1)$$

For this exercise you only need to process bands 1-5 and 7, which are the visible, NIR and SWIR bands with a resolution of 30 m. Bands 6 (1 and 2) are within the thermal part of the electromagnetic spectrum and have a resolution of 60 m. Finally, band 8 is a panchromatic band (measuring from green to NIR) with a spatial resolution of 15 m.

To convert the radiance image to TOA Equation 3.2 needs to be used with the

Table 3.1: Landsat ETM+ (7) gains and offsets for converting DN's to radiance

Band	LMin	LMax	QCal _{min}	QCal _{max}	Wavelength (nm)	Resolution (m)
1	-6.2	191.6	1	255	450–515	30
2	-6.4	196.5	1	255	525–605	30
3	-5.0	152.9	1	255	630–690	30
4	-5.1	157.4	1	255	750–900	30
5	-1.0	31.06	1	255	1550–1750	30
6 (1)	0.0	17.04	1	255	10400–12500	60
6 (2)	3.2	12.65	1	255	10400–12500	60
7	-0.35	10.8	1	255	2090–2350	30
8	-4.7	243.1	1	255	520–900	15

coefficients shown in Table 3.2.

$$\rho = \pi \cdot L_{\lambda} \cdot d^2 \cdot ESUN_{\lambda} \cdot \cos\theta_s \quad (3.2)$$

where, L_{λ} is the radiance measure at the sensor, d is the distance to the sun, $ESUN_{\lambda}$ is the solar irradiance for the wavelength of the image band and θ is the solar zenith.

Table 3.2: Solar irradiance for the Landsat ETM+ bands.

Band	Solar Irradiance ($W \cdot m^{-3}$)
1	1997
2	1812
3	1533
4	1039
5	230.8
7	84.9
8	1362

3.1.1 Convert to Radiance

To convert to the landsat imagery to radiance RSGISLib provides the following command specifically for Landsat processing.

```

1 <!--
2   A command to calibrate image Landsat data from at sensor
3   DNs to at sensor radiance (Eq: ((lmax-lmin)/(qcalmax-qcalmin))
4   * (DNs - qcalmin) + lmin) Eq is from landsat manual.
5 -->
6 <rsgis:command algor="imagecalibration" option="landsatradcal"
7   output="image_out.env" format="GDAL Format" >
8   <rsgis:band name="string" image="image1" band="int"
9     [sensorband="string" |
10    lmin="float" lmax="float" qcalmin="float"
11    qcalmax="float"]/>
12   <rsgis:band name="string" image="image1" band="int"
13     [sensorband="string" |
14    lmin="float" lmax="float" qcalmin="float"
15    qcalmax="float"]/>
16   <rsgis:band name="string" image="image1" band="int"
17     [sensorband="string" |
18    lmin="float" lmax="float" qcalmin="float"
19    qcalmax="float"]/>
20 </rsgis:command>

```

Within the example XML given above the | is symbolising ‘or’ so either a string (as shown below) needs to be specified for the sensor band or the values need to be given. Therefore to convert Landsat data to radiance create an XML file containing the following, making sure the file name is the same as the file you are processing, needs to be created.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     to pre-process Landsat ETM+ imagery.
6     Created by Pete on Wed Nov 28 18:59:38 2012.
7     Copyright (c) 2012 Aber Uni. All rights reserved.
8 -->
9
10 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
11   <rsgis:command algor="imagecalibration" option="landsatradcal"
12     output="L7_20323_20000607_rad.kea" format="KEA" >
13     <rsgis:band name="b1" image="L71203023_02320000617_B10.TIF"
14       band="1" sensorband="LETM7_B1" />

```



```

15     <rsgis:band name="b2" image="L71203023_02320000617_B20.TIF"
16         band="1" sensorband="LETM7_B2" />
17     <rsgis:band name="b3" image="L71203023_02320000617_B30.TIF"
18         band="1" sensorband="LETM7_B3" />
19     <rsgis:band name="b4" image="L71203023_02320000617_B40.TIF"
20         band="1" sensorband="LETM7_B4" />
21     <rsgis:band name="b5" image="L71203023_02320000617_B50.TIF"
22         band="1" sensorband="LETM7_B5" />
23     <rsgis:band name="b7" image="L72203023_02320000617_B70.TIF"
24         band="1" sensorband="LETM7_B7" />
25 </rsgis:command>
26 <rsgis:command algor="imageutils" option="popimgstats"
27     image="L7_20323_20000607_rad.kea" ignore="0"
28     pyramids="yes" />
29 </rsgis:commands>

```

After the radiance calibration command you will notice there is an image utilities command called `popimgstats`. This command will calculate the image band statistics including the image histogram, minimum, maximum, mean, median, mode and standard deviation of the pixel values for each image band. It will also generate image pyramids, which allow fast visualisation of the image within a pyramid aware image viewer.

Running the XML commands

To run this script from a UNIX terminal you should run the `rsgisexe` command from within the same directory as your data (this XML file needs to be saved there as well) and it is executed with the following command:

```
rsgisexe -x PreProcessLandsat.xml
```

3.1.2 Convert to TOA

RSGLib has a command to convert the radiance Landsat image to top of atmosphere (TOA) reflectance. The XML command is shown below, note this also includes the XML for the radiance image as well.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4       XML File for execution within RSGISLib
5       to pre-process Landsat ETM+ imagery.
6       Created by Pete on Wed Nov 28 18:59:38 2012.
7       Copyright (c) 2012 Aber Uni. All rights reserved.
8   -->
9
10 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
11   <rsgis:command algor="imagecalibration" option="landsatradcal"
12     output="L7_20323_20000607_rad.kea" format="KEA" >
13     <rsgis:band name="b1" image="L71203023_02320000617_B10.TIF"
14       band="1" sensorband="LETM7_B1" />
15     <rsgis:band name="b2" image="L71203023_02320000617_B20.TIF"
16       band="1" sensorband="LETM7_B2" />
17     <rsgis:band name="b3" image="L71203023_02320000617_B30.TIF"
18       band="1" sensorband="LETM7_B3" />
19     <rsgis:band name="b4" image="L71203023_02320000617_B40.TIF"
20       band="1" sensorband="LETM7_B4" />
21     <rsgis:band name="b5" image="L71203023_02320000617_B50.TIF"
22       band="1" sensorband="LETM7_B5" />
23     <rsgis:band name="b7" image="L72203023_02320000617_B70.TIF"
24       band="1" sensorband="LETM7_B7" />
25   </rsgis:command>
26   <rsgis:command algor="imageutils" option="popimgstats"
27     image="L7_20323_20000607_rad.kea" ignore="0"
28     pyramids="yes" />
29   <rsgis:command algor="imagecalibration" option="topatmosrefl"
30     input="L7_20323_20000607_rad.kea"
31     output="L7_20323_20000607_toa.kea"
32     format="KEA" scaleFactor="1000"
33     day="17" month="06" year="2000"
34     elevation="57.2241705" datatype="UInt16" >
35     <rsgis:band sensorband="LETM7_B1" />
36     <rsgis:band sensorband="LETM7_B2" />
37     <rsgis:band sensorband="LETM7_B3" />
38     <rsgis:band sensorband="LETM7_B4" />
39     <rsgis:band sensorband="LETM7_B5" />
40     <rsgis:band sensorband="LETM7_B7" />
41   </rsgis:command>
```

```

42     <rsgis:command algor="imageutils" option="poppingstats"
43             image="L7_20323_20000607_toa.kea" ignore="0"
44             pyramids="yes" />
45 </rsgis:commands>

```

The options to note are the use of a scale factor of 1000 (so each reflectance value is multiplied by 1000), giving a range from 0 to 1000 and the use of unsigned 16 bit integers to stored the outputted data. Following processing compare the files sizes of the radiance and TOA images, what do you notice?

To compare the file sizes list the directory using the ‘ls’ command with the ‘-lh’ switches, as shown below.

```
ls -lh
```

Side Question

Why is the unsigned integer 16 bit image so much smaller than the 32 bit floating point image?^a

Is any information / precision lost converting to an unsigned integer rather than using a floating point value?^b

^aA 16 bit value take up half the space of a 32 bit value but integers also compress more efficiently than floating point values as whole values are repeated

^bNo, multiplying by 1000 means that each increment (i.e., value of 1) is equal to 0.1% reflectance and this is well below the noise threshold of the system.

Running the XML commands

To run your commands just add these commands to your radiance XML script and then rerun your script using the rsgisexe command. Both the radiance and the TOA images will be calculated

3.1.3 Re-projecting to OSGB

To reproject the image data to the Ordnance Survey National Grid we need to use the command gdalwarp. GDAL is a software library which allows spatial

located images to read and written where a common interface is provided for all the supported image formats. RSGISLib uses GDAL to read and write images and supports all the image formats that GDAL supports. GDAL also includes a number of command line tools for common tasks such as converting between image formats (`gdal_translate`) and warping images (`gdalwarp`). For more information on GDAL visit the website <http://www.gdal.org>.

To use `gdalwarp` the input and output image coordinate systems and projections need to be specified. There are a number of ways in which this can be done but the preferred method is through the ‘Well-Known Text’ (WKT) format. WKT files for the input projection (UTM 30N WGS84; `utm30wgs83.wkt`) and output projection (OSGB 36; `osgb36.wkt`) have been provided.

To use `gdalwarp` the following options are required (Note, the slash is used to allow the command to be split across multiple lines within the shell script). See the GDAL website for a description of these options - you should get used to consulting online resources to understand how to do what you need to do.

```
gdalwarp -s_srs ./utm30wgs83.wkt -t_srs ./osgb36.wkt -ot UInt16 \  
-wt float32 -srcnodata 0 -order 3 -r cubic -of KEA \  
L7_20323_20000607_toa.kea L7_20323_20000607_toa_osgb.kea
```

3.1.4 Calculating Statistics and Image Pyramids

To generate the image pyramids and image statistics for the warped image generated by `gdalwarp` there is a command `gdalcalcastats`, confusingly this is not part of the GDAL project and has been independently developed but is built on GDAL, hence the name.

```
gdalcalcastats L7_20323_20000607_toa_osgb.kea -ignore 0
```

3.1.5 Calculate an NDVI

To calculate an NDVI the RSGISLib software will again be used. For this a new XML file needs to be generated, to generate a blank XML file the `rsgisexe` command with the `-b` option can be used, as shown below.

```
rsgisexe -b CalcLandsatNDVI.xml
```

Using the new XML file add the following XML to calculate the NDVI using the bandmaths tool.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4       XML File for execution within RSGISLib
5       to calculate the NDVI for a landsat 7 scene
6       Created by Pete on Wed Nov 28 21:11:44 2012.
7       Copyright (c) 2012 Aber Uni. All rights reserved.
8 -->
9
10 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
11   <rsgis:command algor="imagecalc" option="bandmaths"
12     output="L7_20323_20000607_osgb_NDVI.kea" format="KEA"
13     datatype="Float32" expression="(NIR-Red)/(NIR+Red)" >
14     <rsgis:variable name="Red"
15       image="L7_20323_20000607_toa_osgb.kea"
16       band="4" />
17     <rsgis:variable name="NIR"
18       image="L7_20323_20000607_toa_osgb.kea"
19       band="3" />
20   </rsgis:command>
21 </rsgis:commands>

```

3.1.6 Expanding the Processing to Multiple Scenes

To undertake this process on multiple scenes you have been provided with a template script for RSGISLib which contains all the processing stages. Note that the command line processing command has been used to call the gdalwarp command from within the RSGISLib XML script.

```

<!-- A command to execute a command line utilities (e.g., mkdir) -->
<rsgis:command algor="commandline" option="execute" command="string" />

```

The the template is shown below and the key thing to notice is the lack of specified filenames and variables, instead variables start '\$' have been used and will be replaced with the true values at a later stage.

```

1 <rsgis:command algor="imagecalibration" option="landsatradcal"
2     output="$PATH/$FILENAME1_rad.kea" format="KEA" >
3     <rsgis:band name="b1" image="$FILEPATH1"
4         band="1" sensorband="LETM7_B1" />
5     <rsgis:band name="b2" image="$FILEPATH2"
6         band="1" sensorband="LETM7_B2" />
7     <rsgis:band name="b3" image="$FILEPATH3"
8         band="1" sensorband="LETM7_B3" />
9     <rsgis:band name="b4" image="$FILEPATH4"
10        band="1" sensorband="LETM7_B4" />
11    <rsgis:band name="b5" image="$FILEPATH5"
12        band="1" sensorband="LETM7_B5" />
13    <rsgis:band name="b7" image="$FILEPATH6"
14        band="1" sensorband="LETM7_B7" />
15 </rsgis:command>
16 <rsgis:command algor="imageutils" option="popimgstats"
17     image="$PATH/$FILENAME1_rad.kea" ignore="0"
18     pyramids="yes" />
19 <rsgis:command algor="imagecalibration" option="topatmosrefl"
20     input="$PATH/$FILENAME1_rad.kea"
21     output="$PATH/$FILENAME1_toa.kea"
22     format="KEA" scaleFactor="1000"
23     day="$VAR1" month="$VAR2" year="$VAR3"
24     elevation="$VAR4" datatype="UInt16" >
25     <rsgis:band sensorband="LETM7_B1" />
26     <rsgis:band sensorband="LETM7_B2" />
27     <rsgis:band sensorband="LETM7_B3" />
28     <rsgis:band sensorband="LETM7_B4" />
29     <rsgis:band sensorband="LETM7_B5" />
30     <rsgis:band sensorband="LETM7_B7" />
31 </rsgis:command>
32 <rsgis:command algor="imageutils" option="popimgstats"
33     image="$PATH/$FILENAME1_toa.kea" ignore="0"
34     pyramids="yes" />
35 <rsgis:command algor="commandline" option="execute"
36     command="gdalwarp -s_srs ./utm30wgs83.wkt -t_srs
37         ./osgb36.wkt -ot UInt16 -wt float32
38         -srcnodata 0 -order 3 -r cubic -of KEA
39         $PATH/$FILENAME1_toa.kea
40         $PATH/$FILENAME1_toa_osgb.kea" />
41 <rsgis:command algor="imageutils" option="popimgstats"

```

```

42         image="$PATH/$FILENAME1_toa_osgb.kea" ignore="0"
43         pyramids="yes" />
44 <rsgis:command algor="imagecalc" option="bandmaths"
45         output="$PATH/$FILENAME1_osgb_NDVI.kea" format="KEA"
46         datatype="Float32" expression="(NIR-Red)/(NIR+Red)" >
47     <rsgis:variable name="Red"
48         image="$PATH/$FILENAME1_toa_osgb.kea"
49         band="4" />
50     <rsgis:variable name="NIR"
51         image="$PATH/$FILENAME1_toa_osgb.kea"
52         band="3" />
53 </rsgis:command>
54 <rsgis:command algor="imageutils" option="popimgstats"
55         image="$PATH/$FILENAME1_osgb_NDVI.kea" ignore="0"
56         pyramids="yes" />

```

To understand what the different variables have been used for see the list below:

\$PATH - This is the path with in the file system where the output files will be written.

\$FILENAME1 - This is the start of the file name which all output files will have.

\$FILEPATH1 - Band 1 of the input Landsat scene.

\$FILEPATH2 - Band 2 of the input Landsat scene.

\$FILEPATH3 - Band 3 of the input Landsat scene.

\$FILEPATH4 - Band 4 of the input Landsat scene.

\$FILEPATH5 - Band 5 of the input Landsat scene.

\$FILEPATH6 - Band 7 of the input Landsat scene.

\$VAR1 - Day of capture

\$VAR2 - Month of capture

\$VAR3 - Year of capture

\$VAR4 - Solar Elevation


```
16         -v 2000 \  
17         -v 57.2241705  
18  
19 mkdir L7_20323_20020404_Outputs  
20 rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \  
21                 -o LandsatProcessingTemplate_Filenames.xml \  
22                 -p L7_20323_20020404_Outputs \  
23                 -b L7_20323_20020404 \  
24                 -f LE72030232002094EDC00/L71203023_02320020404_B10.TIF \  
25                 -f LE72030232002094EDC00/L71203023_02320020404_B20.TIF \  
26                 -f LE72030232002094EDC00/L71203023_02320020404_B30.TIF \  
27                 -f LE72030232002094EDC00/L71203023_02320020404_B40.TIF \  
28                 -f LE72030232002094EDC00/L71203023_02320020404_B50.TIF \  
29                 -f LE72030232002094EDC00/L72203023_02320020404_B70.TIF  
30 rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \  
31                   -o L7_20323_20020404_PreProcessing.xml \  
32                   -v 4 \  
33                   -v 4 \  
34                   -v 2002 \  
35                   -v 39.9992339  
36  
37 mkdir L7_20323_20020911_Outputs  
38 rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \  
39                 -o LandsatProcessingTemplate_Filenames.xml \  
40                 -p L7_20323_20020911_Outputs \  
41                 -b L7_20323_20020911 \  
42                 -f LE72030232002254SGS00/L71203023_02320020911_B10.TIF \  
43                 -f LE72030232002254SGS00/L71203023_02320020911_B20.TIF \  
44                 -f LE72030232002254SGS00/L71203023_02320020911_B30.TIF \  
45                 -f LE72030232002254SGS00/L71203023_02320020911_B40.TIF \  
46                 -f LE72030232002254SGS00/L71203023_02320020911_B50.TIF \  
47                 -f LE72030232002254SGS00/L72203023_02320020911_B70.TIF  
48 rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \  
49                   -o L7_20323_20020911_PreProcessing.xml \  
50                   -v 11 \  
51                   -v 9 \  
52                   -v 2002 \  
53                   -v 39.1746567  
54  
55 mkdir L7_20323_20030218_Outputs  
56 rsgislibcmdxml.py -i LandsatProcessingTemplate.xml \  

```

```
57         -o LandsatProcessingTemplate_Filenames.xml \  
58     -p L7_20323_20030218_Outputs \  
59     -b L7_20323_20030218 \  
60     -f LE72030232003049SGS00/L71203023_02320030218_B10.TIF \  
61     -f LE72030232003049SGS00/L71203023_02320030218_B20.TIF \  
62     -f LE72030232003049SGS00/L71203023_02320030218_B30.TIF \  
63     -f LE72030232003049SGS00/L71203023_02320030218_B40.TIF \  
64     -f LE72030232003049SGS00/L71203023_02320030218_B50.TIF \  
65     -f LE72030232003049SGS00/L72203023_02320030218_B70.TIF  
66 rsgislibvarsxml.py -i LandsatProcessingTemplate_Filenames.xml \  
67         -o L7_20323_20030218_PreProcessing.xml \  
68         -v 18 \  
69         -v 2 \  
70         -v 2003 \  
71         -v 22.4606180
```

Finally, you run each of the XML scripts generated and using the `rsgisexe` command to process your imagery.

3.2 Linear Spectral Unmixing

RSGISLib provides commands for unconstrained and constrained linear unmixing which is commonly used to understand the proportion of endmembers contributing to the reflectance of each pixel. In the case of linear spectral unmixing then the combination is linear (i.e., additive) and provides the proportion of the reflectance of each input pixel corresponding the endmembers provided. The key to spectral unmixing is the selection of suitable endmembers which need to correspond with the extremes of the feature space of the data you are unmixing. You can define up to $n - 1$ (n is the number of image bands) endmembers but is commonly less.

Using multispectral data within the UK, common endmembers are:

- Photosynthetic Vegetation
- Non-photosynthetic Vegetation
- Shade / Water

Therefore, unmixing for these three parameters will provide an output image with 3 image bands:

1. Proportion of photosynthetic vegetation
2. Proportion of non-photosynthetic vegetation
3. Proportion of shade and water

3.2.1 Defining End Members

There are a number of automated methods for defining end members (see literature) but in this case a manual selection of the end members is undertaken by defining a region (polygon) for each endmember using a shapefile (i.e., using QGIS or ArcMap).

If imagery is well corrected to surface reflectance with a standardised sun and view angle then a common set of endmembers can be used across a set of images but if imagery is poorly corrected (i.e., top of atmospheric reflectance or at sensor radiance) then end members need to be individually defined per image.

Once a set of polygons (one for each endmember has been define) then the following XML command with RSGISLib can be executed to generate a matrix file (.mtxt) where a single spectral profile is calculated for each region and saved as a matrix.

```
<!-- A command to extract the pixel values for
      regions to a matrix file as columns which
      can be used as endmembers for unmixing
-->
<rsgis:command algor="zonalstats" option="endmembers"
               image="image.env" vector="polygons.shp"
               output="output.mtxt" method="polyContainsPixel |
               polyContainsPixelCenter | polyOverlapsPixel |
               polyOverlapsOrContainsPixel | pixelContainsPoly |
               pixelContainsPolyCenter | adaptive | envelope" />
```

So for example:

```

<?xml version="1.0" encoding="UTF-8" ?>
<!--
  Description:
    XML File for execution within RSGISLib
  Created by **ME** on Sat Mar 30 18:12:47 2013.
  Copyright (c) 2013 **Organisation**. All rights reserved.
-->

<rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
  <rsgis:command algor="zonalstats" option="endmembers"
    image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
    vector="./EndMembers.shp" output="./Endmembers"
    method="polyContainsPixelCenter" />
</rsgis:commands>

```

3.2.2 Unmixing the Scene

Using the matrix file representing the end members the scene can be unmixed using either of the following XML commands.

```

<!-- A command to undertake an unconstrained linear spectral
  unmixing of the input image for a set of endmembers -->
<rsgis:command algor="imagecalc" option="unconlinearspecunmix"
  image="image.env" output="image" endmembers="matrix.mtxt"
  [gain="float" offset="float" format="GDAL Format"
  datatype="Byte | UInt16 | Int16 | UInt32 | Int32 |
    Float32 | Float64"] />
<!-- A command to undertake a partially constrained linear spectral
  unmixing of the input image for a set of endmembers where the
  sum of the unmixing will be approximately 1 -->
<rsgis:command algor="imagecalc" option="consum1linearspecunmix"
  image="image.env" output="image" endmembers="matrix.mtxt"
  weight="float" [gain="float" offset="float" format="GDAL Format"
  datatype="Byte | UInt16 | Int16 | UInt32 | Int32 |
    Float32 | Float64"] />

```

The first command is a completely unconstrained approach and will commonly produce results which are unrealistic as the combination (mixture) does not add up to 1 and some values could be negative which is of course impossible (a pixel cannot

be made up of a negative amount of photosynthetic vegetation, for example). The second is partially constrained as the combination (mixture) must add up to 1 but does still allow negative values. There is also a version of least squares (the mathematical method used to solve the unmixing problem) which does not produce negative values (non-negative least squares) but this version is not yet working within RSGISLib.

Therefore, it is recommend that the partially constrained algorithm (which is the implementation ENVI uses) is used as shown below where as the imagery is atmospherically corrected the same endmembers have been used on multiple images.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4       XML File for execution within RSGISLib
5   Created by **ME** on Sat Mar 30 18:12:47 2013.
6   Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8
9 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
10   <rsgis:command algor="zonalstats" option="endmembers"
11       image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
12       vector="./EndMembers.shp" output="./Endmembers"
13       method="polyContainsPixelCenter" />
14   <rsgis:command algor="imagecalc" option="consum1linearspecunmix"
15       image="./L7ETM_530N035W_20100620_AtCor_osgb_masked.kea"
16       output="L7ETM_530N035W_20100620_AtCor_osgb_masked_unmixed.kea"
17       endmembers="./Endmembers.mtxt" weight="35" format="KEA"
18       datatype="Float32" />
19   <rsgis:command algor="imagecalc" option="consum1linearspecunmix"
20       image="./L7ETM_530N035W_20100417_AtCor_osgb_masked.kea"
21       output="L7ETM_530N035W_20100417_AtCor_osgb_masked_unmixed.kea"
22       endmembers="./Endmembers.mtxt" weight="35" format="KEA"
23       datatype="Float32" />
24 </rsgis:commands>

```

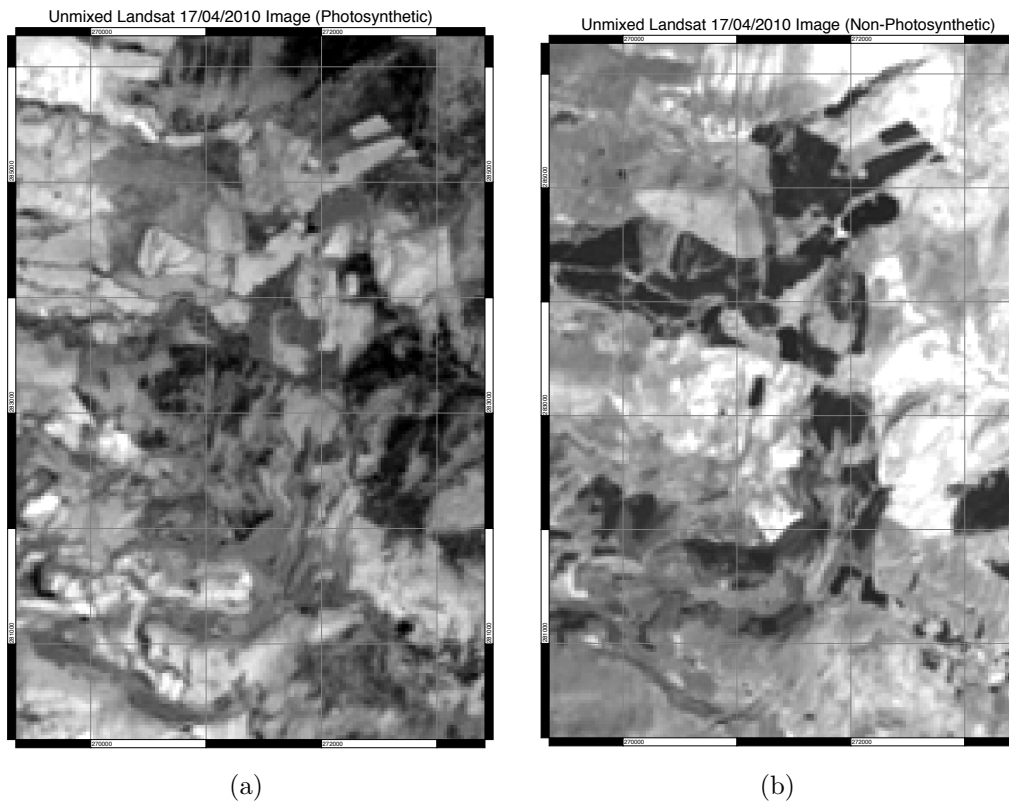


Figure 3.1: Example of a landsat scene which has been unmixed using linear spectral unmixing.

Chapter 4

Examples – Image Utilities

4.1 Creating a new Image

4.1.1 Copying an existing image header

To create a new image with the same dimensions and location and header information as an existing image but with a defined set of image bands, data type and image format the following command can be used. The pixelval option is the value two which all the pixel values will be set to.

```
<rsgis:command algor="imageutils" option="createcopy"  
    image="image.img" output="image_out.env"  
    numbands="int" pixelval="float"  
    format="GDAL Format"  
    datatype="Byte | UInt16 | Int16 |  
    UInt32 | Int32 | Float32 | Float64" />
```

4.2 Editing Header information

4.2.1 Set the projection

The following command allows a ogr wkt file to be used to specify the projection of the image file such that the header is overwritten.

```
<rsgis:command algor="imageutils" option="assignproj"
               image="image.env" projwkt="txt.wkt" />
```

4.2.2 Over write spatial header

The following command allows the spatial header information to be defined. Only the options you wish to specify need to be defined where those which are not defined will not be changed.

```
<rsgis:command algor="imageutils" option="assignspatialinfo"
               image="image.env"
               [tlx="double"] [tly="double"]
               [resX="double"] [resY="double"]
               [rotX="double"] [rotY="double"] />
```

4.2.3 Remove Spatial Header

The following command removes the spatial header from the input image and makes a copy of the input image file.

```
<rsgis:command algor="imageutils" option="removespatialref"
               image="image.env" output="image_out.env" />
```

4.3 Stacking Image Bands

To stack image bands within RSGISLib there are two commands provided, the first attempts to stack all the images (with a specified file extension) within

a directory while the second stacks a specified list of images but in both cases all the images need to intersect and have the same image resolution.

```

1  <!--
2      Stacks all the image bands within a directory into a
3      single image file
4  -->
5  <rsgis:command algor="stackbands" option="dir" dir="input_DIR"
6      output="outputimage" ext="file_extension"
7      format="GDAL Format" datatype="Byte | UInt16 |
8      Int16 | UInt32 | Int32 | Float32 | Float64" />
9  <!--
10     Stacks all the image bands provided in list
11     into a single image file
12 -->
13 <rsgis:command algor="stackbands" option="imgs" output="outputimage"
14     format="GDAL Format" datatype="Byte | UInt16 | Int16 |
15     UInt32 | Int32 | Float32 | Float64" skipValue="float" >
16     <rsgis:image name="band(s) name" file="image1" />
17     <rsgis:image name="band(s) name" file="image2" />
18     <rsgis:image name="band(s) name" file="image3" />
19     <rsgis:image name="band(s) name" file="image4" />
20 </rsgis:command>

```

4.4 Sub-setting Images

RSGISLib provides a number of commands for sub-setting imagery to existing datasets and regions.

4.4.1 Subset to Image

The first command allows an image to be sub-setted to another image (i.e., the region of interest; ROI). Where the output image format and data type can be specified. If they are not specified the default in the ENVI image format and the data type is Float32.

```
<rsgis:command algor="imageutils" option="subset2img"
  image="image.env" output="output_img.env"
  roi="roi.env" format="GDAL Format"
  datatype="Byte | UInt16 | Int16 | UInt32 |
  Int32 | Float32 | Float64" />
```

4.4.2 Subset to Vector

Another option is to subset the input image to the bounding box of a shapefile, this is the region defined by all the geometries within the shapefile.

```
<rsgis:command algor="imageutils" option="subset"
  image="image.env" output="output_img.env"
  vector="vector.shp" format="GDAL Format"
  datatype="Byte | UInt16 | Int16 | UInt32 |
  Int32 | Float32 | Float64" />
```

4.4.3 Subset to Polygons

The final option is to subset the image to a number of output images, one for each polygon within the inputted shapefile. The shapefile attribute tables needs an attribute containing a file path name which will be used to differentiate the output files. The output is the base file path and file name to which the value in the attribute table will be appended and the file extension provided appended on the end.

```
<rsgis:command algor="imageutils" option="subset2polys"
  image="image.env" output="image_out_base"
  vector="vector.shp" outfilename="attribute"
  format="GDAL Format" datatype="Byte | UInt16 |
  Int16 | UInt32 | Int32 | Float32 | Float64"
  extension="env | kea | tif | etc"/>
```

4.5 Generate Image Tiles

4.5.1 Square tiles

To cut an image into a set of image tiles the following command can be used, where the width and height is specified in pixels and an overlap is optional (set to zero if overlap is not required). If the offset option is set, then the files are started half way through a tile (i.e., the first tiles will be half tiles) meaning two tilings can be generated - useful for merging over tile boundaries.

```
<rsgis:command algor="imageutils" option="createtiles"
  image="image.env" output="image_out_base"
  width="int" height="int" overlap="int"
  format="GDAL Format"
  datatype="Byte | UInt16 | Int16 |
            UInt32 | Int32 | Float32 | Float64"
  offset="yes|no" />
```

4.6 Mosaic

The mosaic command in RSGISLib provides a number of options, the simplest (and fastest) method for generating a mosaic assumes square tiles, with no regions containing no data. Images may be specified in the XML or all files in a specified directory with a given extension used.

```
<!-- Mosaic a list of images -->
<rsgis:command algor="imageutils" option="mosaic"
  output="mosaic.kea" nodata="float=0"
  proj="IMAGE"
  format="GDAL Format"
  datatype="Byte | UInt16 | Int16 |   UInt32 |
  Int32 | Float32 | Float64" >
  <rsgis:image file="image1" />
  <rsgis:image file="image2" />
  <rsgis:image file="image3" />
  <rsgis:image file="image4" />
</rsgis:command>
```

```

<!-- Mosaic all images in a directory, with a given extension. -->
<rsgis:command algor="imageutils" option="mosaic"
  output="mosaic.kea" nodata="float=0"
  dir="directory" ext=".ext"
  proj="IMAGE"
  format="GDAL Format"
  datatype="Byte | UInt16 | Int16 |   UInt32 |
  Int32 | Float32 | Float64" />
</rsgis:command>

```

The background value of the new mosaic is set using `nodata`, the default value is 0. As well as the simple mosaic there are a number of options for dealing with overlapping pixels:

skipValue Skip pixels with this value.

skipLowerThresh / skipUpperThresh Skip pixels between upper and lower thresholds, can't be used in combination with skip value.

setSkipBand Band to check for skip value / upper lower threshold. Values are only checked from a single band not all bands. This allows metadata bands (e.g., image date) to be included in the mosaic.

overlapBehaviour Behaviour for overlapping regions

4.7 Sub-Sample the Image file

The following command allows a sample of an image file to be extracted as a HDF5 file, where pixel values with the `nodata` value specified are ignored and only a sample of the pixel values are exported. If a subsample value of 1 is used then every pixel is exported which a value of 2 means every other pixel and a value of 10 means every 10th pixel.

```

<rsgis:command algor="imageutils" option="subsampleimage"
  image="image.kea" output="data.hdf"
  nodata="float" subsample="int" />

```

The following python script can be used to read into the HDF5 file using h5py and plots two columns as a scatter plot.

```
1  #!/usr/bin/env python
2
3  import numpy as np
4  import h5py
5  import sys
6  import matplotlib.pyplot as plt
7
8  def plot(data, x, y):
9      fig = plt.figure()
10     ax = fig.add_subplot(111)
11     ax.scatter(data[...x], data[...y])
12     plt.show()
13
14 def run():
15     file = 'OriginalMoscaic_sample.hdf'
16     if not h5py.is_hdf5(file):
17         print "It was not a hdf5 file."
18         sys.exit()
19
20     f = h5py.File(file, 'r')
21     dset = f['/DATA/DATA']
22     plot(dset, 1, 2)
23     f.close()
24
25 if __name__ == '__main__':
26     run()
```

Chapter 5

Examples – Vectors

5.1 Zonal Statistics

To calculate statistics for pixels falling within a polygon the zonal statistics command in RSGISLib may be used

The main utility for performing zonal stats is pixel stats.

```
1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2 <!--
3     Extract statistics from pixels falling within each polygon in
4     a shapefile.
5 -->
6
7 <rsgis:command algor="zonalstats" option="pixelstats" image="image.env"
8     vector="polys.shp" raster="polys.env"
9     output="output.shp"
10    method="polyContainsPixelCenter"
11    force="yes | no" copyAttributes="yes | no"
12    useBandNames="yes | no"
13    pxlcount="yes | no" mean="yes | no " min="yes | no" max="yes | no "
14    stddev="yes | no" count="yes | no"
15    mode="yes | no" sum="yes | no"
16    minThreshold="float" maxThreshold="float" >
17     <rsgis:attribute name="attribute_name"
18         mean="yes | no " min="yes | no"
```

```

19             max="yes | no " stddev="yes | no" count="yes | no"
20             mode="yes | no" sum="yes | no" >
21     <rsgis:band band="int" minThreshold="float" maxThreshold="float" />
22 </rsgis:attribute>
23 <rsgis:attribute name="attribute_name"
24             mean="yes | no " min="yes | no"
25             max="yes | no " stddev="yes | no" count="yes | no"
26             mode="yes | no" sum="yes | no" >
27     <rsgis:band band="int" minThreshold="float" maxThreshold="float" />
28 </rsgis:attribute>
29 </rsgis:command>
30 </rsgis:commands>

```

image The input image from which statistics will be calculated.

vector OGR file containing polygons for which statistics are to be calculated within. It may extend beyond the polygons but must be in the same projection.

output / outputCSV Shapefile or CSV containing output statistics.

raster Optional. A rasterised version of the input polygons with pixel valued corresponding to the FID of each polygon. Must be the same resolution as the input image. For large datasets using a rasterised version of the shapefile can speed up processing, particularly when the same shapefile is to be used for multiple images.

method The method used to calculate if a pixel is counted as being within a polygon. For options see below. can't be used with raster.

force To overwrite an existing output polygon set "force" to yes.

copyAttributes To copy the attributes from the input file into the output shapefile (in addition to fields created to hold the zonal stats) set "copyAttributes" to yes. This only works for outputting a shapefile, not a CSV.

Pixel in polygon method

A number of methods are available for deciding if a pixel is included within a polygon

- polyContainsPixel - Polygon completely contains pixel
- polyContainsPixelCenter - Pixel centre is within the polygon (default)
- polyOverlapsPixel - Polygon overlaps the pixel
- pixelContainsPoly - Pixel contains the polygon
- pixelContainsPolyCenter - Polygon centre is within pixel
- envelope - All pixels in polygon envelope chosen

If no method is set "polyContainsPixelCenter" will be used.

Attributes

The attribute, the pixel values correspond to (e.g., NDVI, sigma0) are specified using the 'rsgis:attribute' tag, and the image band is defined using the 'rsgis:band' tag.

```

1 <rsgis:attribute name="attribute_name" >
2     <rsgis:band band="int" />
3 </rsgis:attribute>

```

If no attributes are supplied statistics will be calculated for all image bands, named using the band names for the image (if 'useBandNames="yes"') or b1, b2. . . , etc., if band names aren't specified or set in the input image.

Setting minimum and maximum pixel values

Minimum and maximum pixel values may be specified and only pixels falling within these will be used to calculate statistics from, pixels with a value of NaN will automatically be ignored. Min / max values may be defined in the main tag,

so that the same thresholds are applied to every band, or in the band tag to apply only to that band.

```

1 <rsgis:command algor="zonalstats" option="pixelstats" ....
2     minThreshold="float" maxThreshold="float" >
3         <rsgis:attribute name="attribute_name" >
4             <rsgis:band band="int" />
5         </rsgis:attribute>
6 </rsgis:command>

1 <rsgis:command algor="zonalstats" option="pixelstats" ... >
2     <rsgis:attribute name="attribute_name" >
3         <rsgis:band band="int" minThreshold="float" maxThreshold="float" />
4     </rsgis:attribute>
5 </rsgis:command>

```

Available statistics

The statistics which may be calculated from each band are.

- min
- max
- mean
- stdev
- mode - for integer images, e.g., classifications
- sum
- count - this count the number of pixels within the thresholds. It is different from 'pxlcount' which counts the number of pixels within the polygon, which is the same for all attributes.

They may be set using stat="yes" for example to calculate the mean, mean="yes" would be set. Setting in the main tag will apply to all attributes or setting in the attribute tag will only apply to that attribute.

There is a command that only calculates the mean 'pixelmean' this has a smaller memory footprint for very large polygons and has an option if the data is in dB

to ensure correct averaging. For general use the `pixelstats` command is recommended.

5.1.1 Pixel-weighted mean

Rather than using a boolean inclusion / exclusion criteria for determining which pixels to include in the average, the ‘`pixelWeightedMean`’ command uses the relative area of each pixel within the polygon to calculate a weighed average.

$$\bar{x} = \frac{\sum x_i \alpha}{\sum \alpha} \quad (5.1)$$

Where α is the percent of the pixel that intersects the polygon, for example when the polygon is completely contained within the pixel $\alpha = 1$.

5.1.2 `pixelmeanLSSVar`

This is a variation on pixel mean where the pixels are shifted around in a window and the mean on the variance is calculated.

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2   <rsgis:command algor="zonalstats" option="pixelmeanLSSVar"
3     image="image.env" vector="polys.shp"
4     method="polyContainsPixelCenter"
5     output="output.shp" windowSize="int" offsetSize="float"
6     force="yes | no" pxlcount="yes | no">
7     <rsgis:attribute name="attribute_name" >
8       <rsgis:band band="int" threshold="float" />
9     </rsgis:attribute>
10    <rsgis:attribute name="attribute_name" >
11      <rsgis:band band="int" threshold="float" />
12    </rsgis:attribute>
13    <rsgis:attribute name="attribute_name" >
14      <rsgis:band band="int" threshold="float" />
15    </rsgis:attribute>
16  </rsgis:command>
17 </rsgis:commands>

```

Where, `windowSize` is the size of the window to take variance over, e.g., setting at 3 would calculate variance over a 3×3 window. The `offsetSize` is the distance to move the pixels each time. The default of 1 when used with a window size of 3×3 would average over a 3×3 pixel window. Setting an `offsetSize` of 0.5 a window size of 6×6 would average over the same area but at a higher resolution.

5.1.3 Statistics for points

If statistics are required for a point file, two options are available; a) buffer each point and extract statistics from a number of pixels around the point using ‘`pixelstats`’ or b) extract the value for the pixel each point is within using ‘`pointvalue`’.

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2 <!--
3     Buffer each point by a set distance (in coordinate system
4     of shapefile) prior to running pixelstats.
5 -->
6
7     <rsgis:command algor="vectorutils" option="buffervector"
8     vector="/data/inPoints.shp"
9     output="/data/inPoints_buffer.shp"
10    buffer="float" force="yes"/>
11
12
13 </rsgis:commands>

```

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2 <!--
3     Extract the pixel value for each point.
4 -->
5
6 <rsgis:command algor="zonalstats" option="pointvalue"
7     image="image.kea"
8     vector="/data/points.shp"
9     output="/data/output.shp"
10    force="yes | no"
11    useBandNames="yes | no"/>
12
13 </rsgis:commands>

```

As with ‘pixelstats’ a CSV may be outputted instead of a shapefile using ‘outputCSV’ instead of ‘output’

5.2 Image Footprints

This utility will create a shapefile with quadrangles from a CSV file providing the corner coordinates. It was designed for producing a Shapefile with image footprints.

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2 <!-- Create quadrangles based on CSV file -->
3   <rsgis:command algor="vectorutils" option="polygonImageFootprints"
4     input="input.csv" output="vector_out.shp" force="yes | no" />
5 </rsgis:commands>

```

input A comma separated file in the following format (with no header). The scene name must be under 50 characters.

output A shapefile containing the image footprints. Each polygon is attributed with the string in the first column.

force Remove output if it exists.

Table 5.1: Input file format for polygonImageFootprints

Scene Name	UL_X	UL_Y	UR_X	UR_Y	LR_X	LR_Y	LL_X	LL_Y
Scene1	145.667	-28.579	146.315	-28.425	146.473	-28.923	145.822	-29.079
Scene2	144.932	-26.122	145.567	-25.971	145.717	-26.471	145.08	-26.622
Scene3	144.79	-25.629	145.423	-25.479	145.569	-25.979	144.934	-26.131

5.3 Vector Maths

Performs mathematical expressions on attributes of a shapefile. The muparser library is used to input the expression which supports a number of mathematical and binary expressions. Variables are defined using the `rsgis:variable` tag and the fieldname in the attribute table. The output is a shapefile with the result of

the expression (for each polygon) saved in a new column. The heading of the new column is specified using the 'outHeading' tag.

```
1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2     <!-- Performs vector maths on an input shapefile -->
3     <rsgis:command algor="vectorutils" option="vectormaths"
4     input="string" output="string" expression="string"
5     outHeading="string" force="yes | no">
6     <rsgis:variable name="string" fieldname="string"/>
7     <rsgis:variable name="string" fieldname="string"/>
8 </rsgis:command>
```

Chapter 6

Examples - Image Segmentation

6.1 Iterative Elimination Algorithm

The segmentation algorithm (Shepherd et al., 2013) is based on generating spectrally similar units with a minimum object size.

The algorithm consists of a number of steps

1. Select image bands and stack images
2. Stretch image data
3. Find unique cluster within feature space (KMeans)
4. Assign pixels to clusters
5. Clump the image
6. Eliminate small segments

The KMeans clusters takes just a single image where all the bands are used as input so if multiple images are required to be inputted then they need to be stacked and the bands which are to be used selected. As a Euclidean distance is used within

the feature space the image is stretched such that all the pixel values are within the same range (i.e., 0–255).

A clustering algorithm is then used to identify the unique colours within the image, in this case a KMeans clustering is used but other clustering algorithms could also be used instead. The image pixels are then assigned to the clusters (classifying the image) and the image clumped to find the connected regions of the image.

The final step is an iterative elimination of the small segments, starting with the single pixels and going up to the maximum size of the segments specified by the user.

Therefore, there are two key parameters within the algorithm:

1. the number of cluster centres identified by the KMeans clustering
2. the minimum size of the segments

6.1.1 XML Code

```

1 <rsgis:command algor="imageutils" option="stretch" image="$FILEPATH"
2     output="$PATH/$FILENAME_stretched.kea" ignorezeros="yes"
3     stretch="LinearStdDev" stddev="2" format="KEA" />
4
5 <rsgis:command algor="imagecalc" option="bandmaths" output="$PATH/$FILENAME_mask.kea"
6     format="KEA" expression="b1==0?0:1" >
7     <rsgis:variable name="b1" image="$FILEPATH" band="1" />
8 </rsgis:command>
9
10 <rsgis:command algor="imageutils" option="mask"
11     image="$PATH/$FILENAME_stretched.kea"
12     mask="$PATH/$FILENAME_mask.kea"
13     output="$PATH/$FILENAME_stretched_masked.kea"
14     maskvalue="0" outputvalue="0" format="KEA" />
15
16 <rsgis:command algor="commandline" option="execute"
17     command="rm $PATH/$FILENAME_mask.kea" />
18 <rsgis:command algor="commandline" option="execute"
19     command="rm $PATH/$FILENAME_stretched.kea" />
20

```

```
21 <rsgis:command algor="imagecalc" option="kmeanscentres"  
22     image="$PATH/$FILENAME_stretched_masked.kea"  
23     output="$PATH/$FILENAME_clusters" numclusters="60" maxiterations="200"  
24     degreeofchange="0.25" subsample="1" initmethod="diagonal_range_attach" />  
25  
26 <rsgis:command algor="segmentation" option="labelsfromclusters"  
27     image="$PATH/$FILENAME_stretched_masked.kea"  
28     output="$PATH/$FILENAME_clusters.kea"  
29     clusters="$PATH/$FILENAME_clusters.gmtxt"  
30     ignorezeros="yes" format="KEA" proj="IMAGE" />  
31  
32 <rsgis:command algor="segmentation" option="elimsinglepxls"  
33     image="$PATH/$FILENAME_stretched_masked.kea"  
34     clumps="$PATH/$FILENAME_clusters.kea"  
35     temp="$PATH/$FILENAME_clusters_singlepxls_tmp.kea"  
36     output="$PATH/$FILENAME_clusters_nosinglepxls.kea"  
37     ignorezeros="yes" format="KEA" proj="IMAGE" />  
38  
39 <rsgis:command algor="commandline" option="execute"  
40     command="rm $PATH/$FILENAME_clusters.kea" />  
41 <rsgis:command algor="commandline" option="execute"  
42     command="rm $PATH/$FILENAME_clusters_singlepxls_tmp.kea" />  
43  
44 <rsgis:command algor="segmentation" option="clump"  
45     image="$PATH/$FILENAME_clusters_nosinglepxls.kea"  
46     output="$PATH/$FILENAME_clumps.kea" nodata="0"  
47     format="KEA" inmemory="no" proj="IMAGE" />  
48  
49 <rsgis:command algor="commandline" option="execute"  
50     command="rm $PATH/$FILENAME_clusters_nosinglepxls.kea" />  
51  
52 <rsgis:command algor="segmentation" option="rmsmallclumpsstepwise"  
53     image="$PATH/$FILENAME_stretched_masked.kea"  
54     clumps="$PATH/$FILENAME_clumps.kea"  
55     output="$PATH/$FILENAME_clumps_elim.kea"  
56     minsize="50" maxspectraldist="200000"  
57     format="KEA" inmemory="no" proj="IMAGE" />  
58  
59 <rsgis:command algor="commandline" option="execute"  
60     command="rm $PATH/$FILENAME_stretched_masked.kea" />  
61 <rsgis:command algor="commandline" option="execute"
```



```

62         command="rm $PATH/$FILENAME_clumps.kea" />
63
64 <rsgis:command algor="segmentation" option="relabelclumps"
65         image="$PATH/$FILENAME_clumps_elim.kea"
66         output="$PATH/$FILENAME_clumps_elim_final.kea"
67         format="KEA" inmemory="no" proj="IMAGE" />
68
69 <rsgis:command algor="commandline" option="execute"
70         command="rm $PATH/$FILENAME_clumps_elim.kea" />
71
72 <rsgis:command algor="segmentation" option="meaning"
73         image="$FILEPATH" clumps="$PATH/$FILENAME_clumps_elim_final.kea"
74         output="$PATH/$FILENAME_clumps_elim_mean.kea"
75         format="KEA" inmemory="no" proj="IMAGE" />
76
77 <rsgis:command algor="imageutils" option="popimgstats"
78         image="$PATH/$FILENAME_clumps_elim_mean.kea" ignore="0" pyramids="yes" />

```

To use the script provided you need to use the `rsgislibxml.py` command which replaces the `$FILEPATH` with the file path of the input image (found by `rsgislibxml.py` within the input directory) `$PATH` with the provided directory path and `$FILENAME` with the name of the input file. An example of this command is given below:

```

rsgislibxml.py -i RunSegmentationTemplate.xml \
               -o Segmentation.xml -p ./Segments \
               -d ./Data/ -e .kea -r no -t single

```

Once the command above has been executed then the segmentation can be run using the `rsgisexe` command:

```

rsgisexe -x Segmentation.xml

```

The resulting segmentation will have produced 3 output files

1. `*clusters.gmtxt` – Cluster centres.
2. `*clumps_elim_final.kea` – Segment clumps.
3. `*clumps_elim_mean.kea` – Mean colour image using segments.

Following the segmentation the it is recommend that you make sure that the clumps file is defined as a thematic file, as demonstrated in the following piece of

python:

```
1  #!/usr/bin/env python
2
3  import sys
4  from osgeo import gdal
5
6  ds = gdal.Open(sys.argv[1], gdal.GA_Update)
7  for bandnum in range(ds.RasterCount):
8      band = ds.GetRasterBand(bandnum + 1)
9      band.SetMetadataItem('LAYER_TYPE', 'thematic')
```

Finally, use the `gdalcalcstats` command to populate the image with an attribute table, histogram and colour table (set `-ignore 0` as 0 is the background no data value).

```
setthematic.py L7ETM_530N035W_clumps_elim_final.kea
gdalcalcstats L7ETM_530N035W_clumps_elim_final.kea -ignore 0
```

Chapter 7

Examples – Raster GIS

7.1 Populating Clumps with Statistics

7.1.1 Basic Statistics

To populate the segments with statistics (i.e., Mean for each spectral band) there is a command with the `rastergis` part of the RSGISLib software. Examples of this are shown within the XML code below, note the text given for each band is the names of the output columns.

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Thu Mar 21 09:25:21 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8
9 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
10
11   <rsgis:command algor="rastergis" option="popattributestats"
12     clumps="L7ETM_530N035W_Classification.kea"
13     input="L7ETM_530N035W_20100417_AtCor_osgb_masked.kea" >
14     <rsgis:band band="1" mean="MayBlue" stddev="MaySDBlue" />
15     <rsgis:band band="2" mean="MayGreen" stddev="MaySDGreen" />
```

```

16     <rsgis:band band="3" mean="MayRed" stddev="MaySDRed" />
17     <rsgis:band band="4" mean="MayNIR" stddev="MaySDNIR" />
18     <rsgis:band band="5" mean="MaySWIR1" stddev="MaySDSWIR1" />
19     <rsgis:band band="6" mean="MaySWIR2" stddev="MaySDSWIR2" />
20 </rsgis:command>
21
22 <rsgis:command algor="rastergis" option="popattributestats"
23     clumps="L7ETM_530N035W_Classification.kea"
24     input="L7ETM_530N035W_20100620_AtCor_osgb_masked.kea" >
25     <rsgis:band band="1" mean="JuneBlue" stddev="JuneSDBlue" />
26     <rsgis:band band="2" mean="JuneGreen" stddev="JuneSDGreen" />
27     <rsgis:band band="3" mean="JuneRed" stddev="JuneSDRed" />
28     <rsgis:band band="4" mean="JuneNIR" stddev="JuneSDNIR" />
29     <rsgis:band band="5" mean="JuneSWIR1" stddev="JuneSDSWIR1" />
30     <rsgis:band band="6" mean="JuneSWIR2" stddev="JuneSDSWIR2" />
31 </rsgis:command>
32
33 <rsgis:command algor="rastergis" option="popattributestats"
34     clumps="L7ETM_530N035W_Classification.kea"
35     input="Nant_y_Arian_DEM_30m.kea" >
36     <rsgis:band band="1" min="MinDEM" mean="MaxDEM"
37         mean="MeanDEM" stddev="StdDevDEM" />
38 </rsgis:command>
39
40 </rsgis:commands>

```

7.1.2 Calculating Indices

If you are going to use indices and other derived information within your classification it is quite often a good idea to set up a python script to calculate those indices and write them back to the image rather than over complicating your classification script. An example of this is shown below.

```

1  #!/usr/bin/env python
2
3  import sys
4  from rios import rat
5  import numpy
6  import osgeo.gdal as gdal

```

```
7
8
9  #Input file.
10 fname = "L7ETM_530N035W_Classification.kea"
11 ratDataset = gdal.Open( fname, gdal.GA_Update )
12
13 print("Import Columns.")
14 MayBlue = rat.readColumn(ratDataset, "MayBlue")
15 MayGreen = rat.readColumn(ratDataset, "MayGreen")
16 MayRed = rat.readColumn(ratDataset, "MayRed")
17 MayNIR = rat.readColumn(ratDataset, "MayNIR")
18 MaySWIR1 = rat.readColumn(ratDataset, "MaySWIR1")
19 MaySWIR2 = rat.readColumn(ratDataset, "MaySWIR2")
20
21 JuneBlue = rat.readColumn(ratDataset, "JuneBlue")
22 JuneGreen = rat.readColumn(ratDataset, "JuneGreen")
23 JuneRed = rat.readColumn(ratDataset, "JuneRed")
24 JuneNIR = rat.readColumn(ratDataset, "JuneNIR")
25 JuneSWIR1 = rat.readColumn(ratDataset, "JuneSWIR1")
26 JuneSWIR2 = rat.readColumn(ratDataset, "JuneSWIR2")
27
28 MeanDEM = rat.readColumn(ratDataset, "MeanDEM")
29
30 MayNIR.astype(numpy.float32)
31 MayRed.astype(numpy.float32)
32 JuneNIR.astype(numpy.float32)
33 JuneRed.astype(numpy.float32)
34 MayBlue.astype(numpy.float32)
35 JuneBlue.astype(numpy.float32)
36
37 print("Calculate Indices.")
38 MayNDVI = (MayNIR - MayRed) / (MayNIR + MayRed)
39 JuneNDVI = (JuneNIR - JuneRed) / (JuneNIR + JuneRed)
40
41 MayWBI = MayBlue/MayNIR
42 JuneWBI = JuneBlue/JuneNIR
43
44 rat.writeColumn(ratDataset, "MayNDVI", MayNDVI)
45 rat.writeColumn(ratDataset, "JuneNDVI", JuneNDVI)
46 rat.writeColumn(ratDataset, "MayWBI", MayWBI)
47 rat.writeColumn(ratDataset, "JuneWBI", JuneWBI)
```

7.1.3 Percentiles: Median

If you want to use a percentile (i.e., median) then the following option is also available (popattributepercentile).

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Fri Apr 5 21:58:44 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9   <rsgis:command algor="rastergis" option="popattributepercentile"
10     clumps="L7ETM_530N035W_Classification.kea"
11     input="L7ETM_530N035W_20100417_AtCor_osgb_masked.kea" >
12   <rsgis:band band="1" name="BlueMedian" percentile="50" />
13   <rsgis:band band="2" name="GreenMedian" percentile="50" />
14     <rsgis:band band="3" name="RedMedian" percentile="50" />
15     <rsgis:band band="1" name="NIRMedian" percentile="50" />
16   <rsgis:band band="2" name="SWIR1Median" percentile="50" />
17     <rsgis:band band="3" name="SWIR2Median" percentile="50" />
18   </rsgis:command>
19 </rsgis:commands>

```

7.1.4 Location

If location of the of the segments is required this can be populated using the following command.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Fri Apr 5 22:12:06 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9   <rsgis:command algor="rastergis" option="spatiallocation"
10     image="L7ETM_530N035W_Classification.kea"

```

```
11         eastings="Eastings" northings="Northings" />
12 </rsgis:commands>
```

7.1.5 Shape Parameters

If parameters on the shape of an object are required the following command can be used where the following features are currently available:

- Area
- Asymmetry
- Border Index
- Border Length
- Compactness
- Density
- Elliptic Fit
- Length
- Length Width Ratio
- Width
- Main Direction
- Radius Largest Enclosed Ellipse
- Radius Smallest Enclosed Ellipse
- Rectangular Fit
- Roundness
- Shape Index

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3     Description:
4     XML File for execution within RSGISLib
```

```

5      Created by **ME** on Fri Apr 5 22:20:48 2013.
6      Copyright (c) 2013 **Organisation**. All rights reserved.
7      -->
8
9      <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
10         <rsgis:command algor="rastergis" option="calcshapeindices"
11             clumps="L7ETM_530N035W_Classification.kea" >
12             <rsgis:index name="Area" column="AreaCol" />
13             <rsgis:index name="BorderLength" column="BorderLenCol" />
14             <rsgis:index name="ShapeIndex" column="ShapeIdxCol" />
15             <rsgis:index name="LengthWidth" column="LenWidCol" />
16         </rsgis:command>
17 </rsgis:commands>

```

7.1.6 Relative Border to a Class

To calculate the relative border of each clump to a clumps with a particular class, for answering questions such as which clumps have a border to urban, the following command has been made available.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3     Description:
4         XML File for execution within RSGISLib
5     Created by **ME** on Fri Apr 5 22:26:24 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7     -->
8
9 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
10     <rsgis:command algor="rastergis" option="calcrelborderlength"
11         clumps="L7ETM_530N035W_Classification.kea"
12         ignorezeroedges="yes" colname="Bord2Urban"
13         classcolumn="Classification"
14         classname="Urban" />
15 </rsgis:commands>

```


7.1.7 Populating with an Existing Classification

Where an existing classification is available and you are looking at classifying change for example the classification can be intersected with a new segmentation (clumps file) using the following command.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4       XML File for execution within RSGISLib
5   Created by **ME** on Fri Apr 5 23:41:18 2013.
6   Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9   <rsgis:command algor="rastergis" option="popcategoryproportions" clumps="clumps.env" categori
10 </rsgis:commands>

```

7.2 Classification

A range of classification options are available and are being added but it is recommended that classification is undertaken through the python scripting language where the attribute table can be accessed using the RIOS library (see Bunting and Clewley (2013) for a tutorial). Numpy ‘where’ statements can be used to implement a rule based classification. Alternatively, if you want to use other supervised and unsupervised classifiers there are implementations within numerous freely available python libraries (see machine learning python (mlpy) or scikit-learn libraries).

7.2.1 Rule Based

A rule based classification can be applied using python see (Bunting and Clewley, 2013) for a tutorial.

7.2.2 Updating an Existing Classification

7.3 Change Detection

A per class change detection technique is available within the library. The algorithm requires a classification to be provided, see the earlier command. The algorithm works by identifying for each class a region of the feature space which is specified as a number of standard deviations from the mean (i.e., ± 3 standard deviations from the mean as shown below), where this can be specified on a per class basis. The columns used for this test are specified within a comma separated list (fields attribute). If different columns should be used for different classes then you need to duplication this command (e.g., different indices are related to different classes). The algorithm will then identify those clumps which are outside of this ‘valid’ range and these are likely to be regions of change or areas of inconstancy with the rest of the class.

Therefore, the command can be used in two way:

1. To check the consistency of an existing map and dataset.
2. To find change from a new image given an old map.

Class names are provided as strings (text), while the ‘changeval’ is the value outputted to the ‘ChangeField’. In the example shown below a different value for the ‘changeval’ was specified for each class retaining explicit information as to its previous class but this value should be set to 1 for all classes to simply create a binary mask of change.

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Fri Apr 5 23:28:34 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9   <rsgis:command algor="rastergis" option="findchangeclumpsfromstddev"
10     clumps="444432_071117_clumps_lcdb.kea"

```

```

11         classcol="LCDB3_Maj" changefield="ChangeFound"
12         fields="MeanGreenRefl,MeanRedRefl,MeanNIRRefl,MeanSWIRRefl" >
13         <rsgis:class name="71" stddevthres="3" changeval="1" />
14         <rsgis:class name="69" stddevthres="3" changeval="2" />
15         <rsgis:class name="68" stddevthres="3" changeval="3" />
16         <rsgis:class name="64" stddevthres="3" changeval="4" />
17         <rsgis:class name="54" stddevthres="3" changeval="5" />
18         <rsgis:class name="52" stddevthres="3" changeval="6" />
19         <rsgis:class name="51" stddevthres="3" changeval="7" />
20         <rsgis:class name="40" stddevthres="3" changeval="8" />
21         <rsgis:class name="41" stddevthres="3" changeval="9" />
22         <rsgis:class name="43" stddevthres="3" changeval="10" />
23     </rsgis:command>
24 </rsgis:commands>

```

7.4 Other Utilities

7.4.1 Export Columns to Raster Bands

A command is available to export individual columns as individual image bands, such as below:

```

1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3     Description:
4         XML File for execution within RSGISLib
5         Created by **ME** on Fri Apr 5 22:39:42 2013.
6         Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9     <rsgis:command algor="rastergis" option="exportcols2raster"
10         clumps="L7ETM_530N035W_Classification.kea"
11         output="L7ETM_530N035W_ClumpMeans.kea"
12         format="KEA" datatype="Float32" >
13         <rsgis:field name="MayBlue" />
14         <rsgis:field name="MayGreen" />
15         <rsgis:field name="MayRed" />
16         <rsgis:field name="MayNIR" />

```

```
17     <rsgis:field name="MaySWIR1" />
18     <rsgis:field name="MaySWIR2" />
19 </rsgis:command>
20 </rsgis:commands>
```

7.4.2 Calculate Statistics

Rather than using the `gdalcalcstats` or the `RSGISLib` command `popimgstats` (within `imageutils`) which calculate information which is not necessarily required for raster GIS application and can be a quite slow for very large clump files this command is specifically written for only populating the statistics and attribute table required for raster GIS applications using the following command:

```
1 <?xml version="1.0" encoding="UTF-8" ?>
2 <!--
3   Description:
4     XML File for execution within RSGISLib
5     Created by **ME** on Fri Apr 5 23:10:29 2013.
6     Copyright (c) 2013 **Organisation**. All rights reserved.
7 -->
8 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
9   <rsgis:command algor="rastergis" option="populatestats"
10     clumps="L7ETM_530N035W_Segmentation.kea"
11     pyramids="yes"
12     colourtable="yes" />
13 </rsgis:commands>
```

Chapter 8

Examples – Image Registration

8.1 Tie Point Generation

```
1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2
3 <!-- A command for automatically generating a set of ground control
4      points between a pair of images. A basic algorithm with a single
5      iteration and movement of gcps -->
6 <rsgis:command algor="registration" option="basic" reference="image"
7      floating="image" output="gcps.txt"
8      outputType="envi_img2img|envi_img2map|rsgis_img2map"
9      metric="euclidean|sqdiff|manhattan|correlation"
10     pixelgap="int" window="int" search="int"
11     threshold="float" stddevRef="float" stddevFloat="float"
12     subpixelresolution="int"/>
13
14 <!-- A command for automatically generating a set of ground control points
15      between a pair of images. An algorithm with a single layer of connected
16      gcps which used IDW to shift neighbouring gcps within a distance threshold -->
17 <rsgis:command algor="registration" option="singlelayer" reference="image"
18     floating="image" output="gcps.txt"
19     outputType="envi_img2img|envi_img2map|rsgis_img2map"
20     metric="euclidean|sqdiff|manhattan|correlation"
21     pixelgap="int" window="int" search="int"
22     threshold="float" stddevRef="float" stddevFloat="float"
```

```
23     subpixelresolution="int" distanceThreshold="float"  
24     maxiterations="int" movementThreshold="float" pSmoothness="float"/>  
25  
26 </rsgis:commands>
```

Basic Algorithm

The basic algorithm treats all the tie point independently and where they are each moved to their optimal place within the search space defined but the metric, the data window and the search space (i.e., maximum distance the tie point can be moved). The parameters for this algorithm are:

reference The reference image which to which the floating image is to be registered.

floating The floating image to be registered to the reference image

output The output file containing the generated tie points.

outputType The format of the output file. The output format supports exporting for use within ENVI or a format defined for the rsgislib warp functions (see below)

metric The similarity metric to be used to compare the images. The recommended default is 'correlation' (particularly for multi-modal imagery) but for images where the pixel values are very similar the euclidean, sqdiff or manhattan metrics can offer equivalent matching performance but are faster to calculate.

pixelgap The gaps in image pixels between the initial tie points (this is for both the x and y axis).

window The size of the window around each tie point which will be used for the matching.

search The distance (in pixels) from the tie point start point which will be searched.

threshold The threshold for the image metric above/below (depending on image metric) which matching is considered insufficient to be reliable and therefore

the match will be ignored.

stddevRef The threshold which defines the standard deviation for the window around each tie point below which it is deemed there is insufficient information to perform a match (note that the tie point window has to be below the threshold for both the reference and floating image to be ignored).

stddevFloat The threshold which defines the standard deviation for the window around each tie point below which it is deemed there is insufficient information to perform a match (note that the tie point window has to be below the threshold for both the reference and floating image to be ignored).

subpixelresolution The sub-pixel resolution to which the pixel shifts are estimated. Note that the values are positive integers such that a value of 2 will result in a sub pixel resolution of 0.5 of a pixel and a value 4 will be 0.25 of a pixel.

Single Layer Algorithm

The single layer algorithm generates a connected grid of tie points where the the inverse weighted distance of the movement of a current tie point is applied to the connected (within a given distance) tie points. The additional parameters for this algorithm are:

distanceThreshold The distance (in pixels) to be connected within the layer.

maxiterations The maximum number of iterations of the tie point grid to find an optimal set of tie points.

movementThreshold The threshold for the average amount of tie point movement for the optimisation to be terminated.

pSmoothness The p parameter for the inverse weighted distance calculation. A value of 2 should be used by default.

Note: If a negative correlation is found is it turned positive for the metric as the correlation is being maximised. For some datasets a strong negative correlation will be expected as they are the inverse of each other.

8.2 Image Warping

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2 <!-- A command to warp an input image using a set of
3      ground control points using a nearest neighbour algorithm -->
4 <rsgis:command algor="registration" option="nnwarp"
5      gcps="string" image="string" output="string" projection="file.wkt"
6      resolution="float" format="string" />
7
8 <!-- A command to warp an input image using a set of
9      ground control points using a triangulation -->
10 <rsgis:command algor="registration" option="triangularwarp"
11      gcps="string" image="string" output="string" projection="file.wkt"
12      resolution="float" format="string" />
13
14 <!-- A command to warp an input image using a set of
15      ground control points using a polynomial -->
16 <rsgis:command algor="registration" option="polywarp"
17      gcps="string" image="string" output="string"
18      projection="file.wkt" resolution="float" format="string" polyOrder="int" />
19
20 <!-- A command to add tie points as GCPs to GDAL dataset -->
21 <rsgis:command algor="registration" option="gcp2gdal" gcps="string"
22      image="image" [output="image" ] />
23 </rsgis:commands>

```

gcps The gcps file as exported from the rsgislib tie point generation commands (above)

image The floating image to be warped

output The file path for the output image

projection A file path to a text file containing the WKT string for the projection of the output image

resolution The pixel resolution of the output image.

format The output image format (See GDAL driver names) default is ENVI if not specified. For erdas imagine file use 'HFA'.

Nearest Neighbour Warping

The nearest neighbour warping algorithm simply transforms the image pixels in the input image using the spatially closest tie point in the gcps file. It is likely that this algorithm will produce poor results when the distance between tie points is quite large. It is also sensitive to errors in tie point locations.

Triangulation Warping

The triangulation warping produces a delaunay triangulation between the tie points and a linear interpolation of the transformation is applied within the triangles to interpolate the individual pixels. This algorithm produces very good results where a dense grid of tie points is provided but is sensitive to noise.

Polynomial Warping

Polynomial warping is a standard approach to warping remotely sensed imagery and is good for correcting systematic errors (such as aligned an unregistered satellite image) but for random errors (such as aircraft movement) the polynomial model cannot represent all the variation within the scene. Where the polynomial model is appropriate it is robust to noise and gives good results.

Export to GDAL GCPs

Add GCPs to a GDAL dataset to warp using `gdalwarp`.

Chapter 9

Examples – Radar (SAR)

9.1 Biomass Estimation

The biomass estimation algorithm of Saatchi et al. (2007) is available in RSGIS-Lib¹. The algorithm uses empirically derived coefficients, which must be derived specific to the study site / time of acquisition. Although SAR sensors don't directly measure biomass, backscatter is a function of the structural and dielectric properties and due to this structural link a number of studies have demonstrated a correlation between backscatter and biomass, with the strength of correlation varying depending on forest type and a number of other parameters.

The above ground biomass (AGB) algorithm of Saatchi et al. (2007) is a semi-empirical approach using statistical regression. However, rather than total AGB, relationships are established between the crown and trunk components separately. The form of the equations, are linked to a simplified backscatter model and thus the algorithm can be considered a semi-empirical approach. The equations are given as:

$$\begin{aligned} \log(B_c) = & a_0 + a_1\sigma_{HV}^0 \cos(\theta_0 - \theta_l) + a_2(\sigma_{HV} \cos(\theta_0 - \theta_l))^2 \\ & + b_1\sigma_{HH}^0 \sin(\theta_0 - \theta_l) + b_2(\sigma_{HH} \sin(\theta_0 - \theta_l))^2 \\ & + c_1\sigma_{VV}^0 \cos(\theta_0 - \theta_l) + c_2(\sigma_{VV} \cos(\theta_0 - \theta_l))^2 \end{aligned} \quad (9.1)$$

¹This section was modified from Chapter 2 of Clewley (2012) and describes work led by R. Lucas on a ESA funded report.

$$\begin{aligned} \log(B_t) = & a_0 + a_1\sigma_{HV}^0 \sin(\theta_0 - \theta_l) + a_2(\sigma_{HV} \sin(\theta_0 - \theta_l))^2 \\ & + b_1\sigma_{HH}^0 \cos(\theta_0 - \theta_l) + b_2(\sigma_{HH} \cos(\theta_0 - \theta_l))^2 \\ & + c_1\sigma_{VV}^0 \cos(\theta_0 - \theta_l) + c_2(\sigma_{VV} \cos(\theta_0 - \theta_l))^2 \end{aligned} \quad (9.2)$$

where B_c and B_t are crown and trunk biomass respectively² and $a - c$ are coefficients. The equation takes a quadratic form to account for the loss of radar sensitivity at higher values of AGB. To correct for variations in backscatter because of changes in incidence angle across the radar swath and topography, the local incidence angle (θ_l) is included in the relationship, such that:

$$\cos \theta_l = \sin \alpha \sin \theta_0 \cos(\beta - \beta_s) + \cos \alpha \cos \theta_0 \quad (9.3)$$

where α is the local slope, β the azimuth angle of the radar illumination direction, β_s the aspect angle at the local slope and θ_0 represents the incidence angle of the SAR platform at the centre of the image pixel.

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2
3   <rsgis:command algo="SaatchiBiomass" option="stem"
4     input="inputimage" output="outputimage"
5     a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
6
7   <rsgis:command algo="SaatchiBiomass" option="crown"
8     input="inputimage" output="outputimage"
9     a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
10 </rsgis:commands>

```

The combination of L- and P-band data may also be used to estimate canopy biomass such that:

$$\begin{aligned} \log(B_c) = & a_0 + a_1\sigma_{LHV}^0 \cos(\theta_0 - \theta_l) + a_2(\sigma_{LHV} \cos(\theta_0 - \theta_l))^2 \\ & + b_1\sigma_{PHV}^0 \cos(\theta_0 - \theta_l) + b_2(\sigma_{PHV} \cos(\theta_0 - \theta_l))^2 \end{aligned} \quad (9.4)$$

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2   <rsgis:command algo="SaatchiBiomass" option="crownPL"
3     input="inputimage" output="outputimage"
4     a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
5 </rsgis:commands>

```

²For consistency, B is used for AGB instead of W as in Saatchi et al. (2007)

Where the topography is very flat or there is no known information on local incidence angle, $\theta_0 - \theta_l$ can be assumed to be zero, simplifying 9.1 and 9.2 to:

$$\log(B_c) = a_0 + a_1\sigma_{HV}^0 + a_2\sigma_{HV}^2 + c_1\sigma_{VV}^0 + c_2\sigma_{VV}^2 \quad (9.5)$$

$$\log(B_t) = \begin{matrix} a_0 + b_1\sigma_{HH}^0 + b_2\sigma_{HH}^2 \\ + c_1\sigma_{VV}^0 + c_2\sigma_{VV}^2 \end{matrix} . \quad (9.6)$$

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2   <!-- Calculate stem biomass from a polarimetric SAR image.
3     No correction for incidence angle is used -->
4   <rsgis:command algor="SaatchiBiomass" option="stemNIA"
5     input="inputimage" output="outputimage"
6     a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
7
8   <!-- Calculate crown biomass from a polarimetric SAR image.
9     No correction for incidence angle is used -->
10  <rsgis:command algor="SaatchiBiomass" option="crownNIA"
11    input="inputimage" output="outputimage"
12    a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
13
14  <!-- Calculate crown biomass from a dual-band polarimetric SAR image.
15    No correction for incidence angle is used -->
16  <rsgis:command algor="SaatchiBiomass" option="crownPLNIA"
17    input="inputimage" output="outputimage"
18    a0="a0" a1="a1" a2="a2" b1="b1" b2="b2" c1="c1" c2="c2"/>
19 </rsgis:commands>

```

Within RSGISLib coefficients are available which were derived from AIRSAR data acquired in 2000 over the Injune Landscape Collaborative Project (ILCP) in central southeast Queensland, Australia and LiDAR derived biomass (Lucas et al., 2006). Total LiDAR-derived AGB was split into canopy and trunk components, based on field data and allometric equations (Lucas et al., 2004). Coefficients are provided in Table 9.1. The best fits were obtained using L-band data (crown and trunk; $r^2=0.89$ in both cases).

Within RSGISLib these coefficients are available using:

```

1 <rsgis:commands xmlns:rsgis="http://www.rsgislib.org/xml/">
2   <rsgis:command algor="SaatchiBiomass" option="openStemL"

```

Table 9.1: Coefficients for the semi-empirical method of biomass estimation (Saatchi et al., 2007)

		Coefficient							
Layer	Band	a_0	a_1	a_2	b_1	b_2	c_1	c_2	r^2
Crown	L	6.64	0.296	3.74×10^{-3}	–	–	-0.24	-9.60×10^{-3}	0.89
	P	2.84	0.108	4.26×10^{-4}	–	–	-0.392	-1.39×10^{-2}	0.84
	L & P	2.28	-0.556	-1.42×10^{-2}	0.306	4.95×10^{-3}	–	–	0.86
Trunk	L	3.27	–	–	0.347	0.010	-0.510	-1.96×10^{-2}	0.89
	P	3.89	–	–	0.18	5.39×10^{-3}	-0.278	-1.32×10^{-2}	0.86

```

3     input="inputimage" output="outputimage" />
4
5     <rsgis:command algor="SaatchiBiomass" option="openStemP"
6         input="inputimage" output="outputimage" />
7
8     <rsgis:command algor="SaatchiBiomass" option="openCrownL"
9         input="inputimage" output="outputimage" />
10
11    <rsgis:command algor="SaatchiBiomass" option="openCrownP"
12        input="inputimage" output="outputimage" />
13
14    <rsgis:command algor="SaatchiBiomass" option="openCrownPL"
15        input="inputimage" output="outputimage" />

```

Chapter 10

Examples – Other Utilities

10.1 Running Command Line Tools from XML

RSGISLib provides a command which allows other command line tools or scripts to be executed from within the XML interface. This could be something as simple as creating a new directory or deleting some temporary files.

```
<rsgis:command algor="commandline" option="execute" command="string" />
```

10.1.1 Creating a Directory

The following command will create an output directory 'OutputFiles'.

```
<rsgis:command algor="commandline" option="execute"  
  command="mkdir OutputFiles" />
```

10.1.2 Deleting Files

The following command would delete all files which start with 'TempImage' and end with '.kea'.

```
<rsgis:command algor="commandline" option="execute"  
  command="rm TempImage*.kea" />
```

Bibliography

- Bunting, P., Clewley, D., 2013. Python scripting for spatial data processing. Online at bitbucket.org/petebunting 1, 1–193.
- Clewley, D., Feb. 2012. Retrieval of Forest Structure and Biomass From Radar Data Using Backscatter Modelling and Inversion. Aberystwyth University, 1–356.
- Lucas, R. M., Cronin, N., Lee, A., Moghaddam, M., Witte, C., Tickle, P., 2006. Empirical relationships between AIRSAR backscatter and LiDAR-derived forest biomass, queensland, Australia. *Remote Sensing of Environment* 100 (3), 407–425.
- Lucas, R. M., Moghaddam, M., Cronin, N., 2004. Microwave scattering from mixed-species forests, Queensland, Australia. *IEEE Transactions on Geoscience and Remote Sensing* 42 (10), 2142–2159.
- Saatchi, S. S., Halligan, K., Despain, D., Crabtree, R., 2007. Estimation of Forest Fuel Load From Radar Remote Sensing. *IEEE Transactions on Geoscience and Remote Sensing* 45 (6), 1726–1740.
- Shepherd, J., Bunting, P., Dymond, J., 2013. Segmentation of imagery based on iterative elimination. *Remote Sensing Accepted for Publication*.