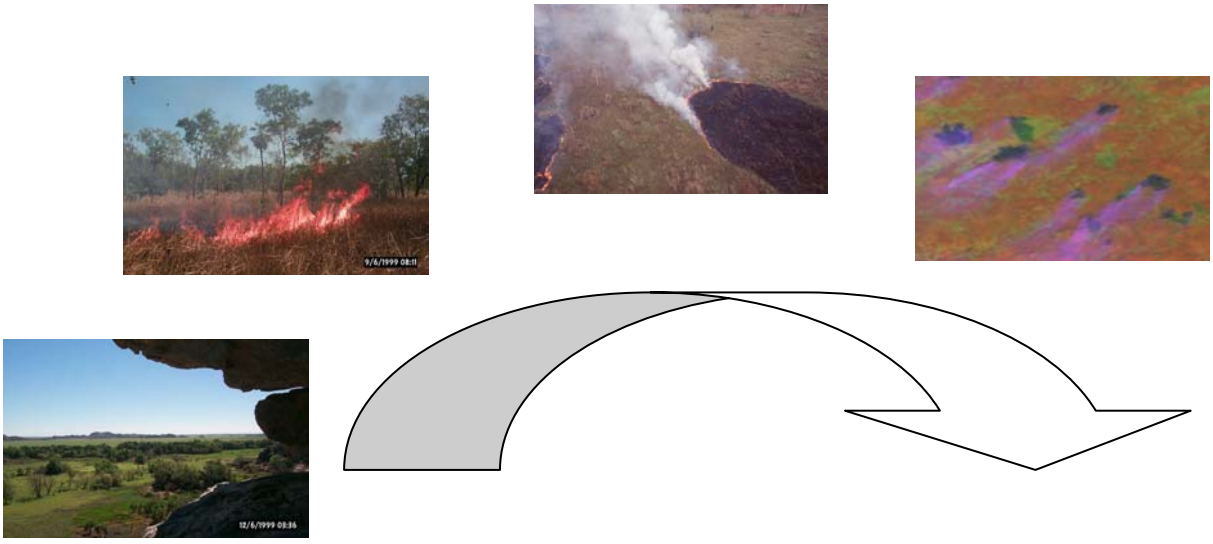
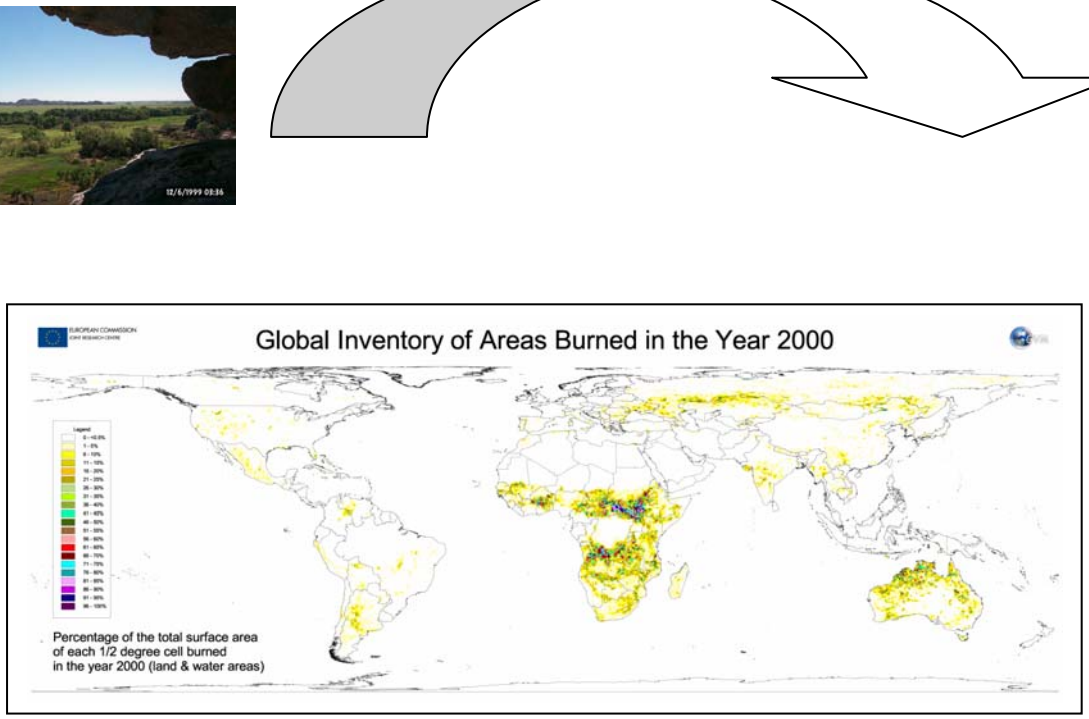
Implementation of Regional Burnt Area Algorithms for the GBA2000 Initiative





Kevin J. Tansey

with contributions from

E. Binaghi, L. Boschetti, P.A. Brivio, A. Cabral, D. Ershov, S. Flasse, R. Fraser, I. Gallo, D. Graetz, J-M. Grégoire, M. Maggi, P. Peduzzi, J.M. Pereira, A. Sà, J. Silva, A. Sousa, D. Stroppiana, M.J.P. Vasconcelos

EUROPEAN COMMISSION
JOINT RESEARCH CENTRE

2002 EUR 20532 EN

EUR 20532 EN

Printed in Italy

Implementation of Regional Burnt Area Algorithms for the GBA2000 Initiative

Kevin J. Tansey[1]

with contributions from

E. Binaghi[2], L. Boschetti[1], P.A. Brivio[3], A. Cabral[4], D. Ershov[5], S. Flasse[6], R. Fraser[7], I. Gallo[2], D. Graetz[8], J-M. Grégoire[1], M. Maggi[1], P. Peduzzi[9], J.M. Pereira[4], A. Sà[10], J. Silva[10], A. Sousa[11], D. Stroppiana[1], M.J.P. Vasconcelos[4]

[1]Global Vegetation Monitoring Unit, Joint Research Centre, Ispra, Italy
[2]Università Degli Studi Dell'Insubria, Varese, Italy
[3]Institute for Electromagnetic Sensing of the Environment, Milan, Italy
[4]Tropical Research Institute, Lisbon, Portugal
[5]International Forest Institute, Moscow, Fed. of Russia
[6]Flasse Consulting, Maidstone, United Kingdom
[7]Canada Centre for Remote Sensing, Ottawa, Canada
[8]CSIRO Earth Observation Centre, Canberra, Australia
[9]UNEP Early Warning Unit, Geneva, Switzerland
[10]Instituto Superior de Agronomia, Lisbon, Portugal
[11]Universidade de Évora, Évora, Portugal

TABLE OF CONTENTS

EXECUTIVE SUMMARY ..... **VIII**

ACRONYMS ..... **X**

1 AN OVERVIEW OF DATA CHARACTERISTICS AND IMAGE PROCESSING **1**

1.1 SPOT Vegetation global S1 data characteristics 1

1.1.1 S1 data format *3*

1.2 Utilisation of global land cover products 4

1.2.1 UMD land cover product data access and characteristics *4*

1.2.2 SPOT Vegetation S1 - UMD land cover data co-registration *6*

1.2.3 Secondary mask products derived from the land cover data *6*

1.3 A simplified overview of the main processing tasks 7

1.4 Computing requirements of GBA2000 8

1.4.1 Automating the processing using c-shell scripts *9*

1.4.2 Location of executable binaries and source code *9*

1.4.3 Computational processing demands geographical area relationships *10*

1.4.4 Further software requirements *11*

1.5 Utilisation of digital elevation model (DEM) data 12

1.5.1 Determining sun shadow affected pixels *13*

1.5.2 Sun shadow masking procedure *14*

2 DATA EXTRACTION FROM TAPE ARCHIVE **16**

2.1 Data format of original SPOT Vegetation S1 archived products 16

2.2 Controlling the time period of analysis: input_dates.txt 18

2.3 Extracting SPOT Vegetation S1 data from tape archive 19

2.4 Simultaneous extraction of registered UMD land cover information 26

2.5 Special consideration for specific GBA2000 data extractions 28

2.6 Missing or problematic data on the tape archive 29

3 GENERIC PRE-PROCESSING MODULE **30**

3.1 Masking of pixels acquired at extreme viewing zenith angles 33

3.2 Masking of pixels affected by saturation in the MIR channel 35

3.3 Masking of cloudy pixels 36

3.4 Masking of cloud shadow pixels 39

3.5 Masking for water and non-vegetated land surfaces 44

3.6 Pre-processing module output product description 45

4 IMAGE DATA COMPOSITING MODULE 49

4.1 MinNIR compositing method and programs 49

4.1.1 Comparison of pre-processing S1 and original S1 composites 53

4.2 MaxNDVI compositing method and programs 55

4.3 NIR value compositing method and programs 57

5 IFI ALGORITHM IMPLEMENTATION MODULE 59

5.1 IFI data extraction from tape archive 61

5.2 IFI pre-processing procedures 62

5.3 IFI burnt area algorithm procedure 68

5.4 IFI post-processing (stage 1) procedure 72

5.5 IFI post-processing (stage 2) procedure 74

6 UTL ALGORITHM IMPLEMENTATION MODULE 81

6.1 UTL Africa 1 module 81

6.1.1 UTL pre-processing procedure for all UTL algorithms 82

6.1.2 UTL compositing procedure for all UTL algorithms 83

6.1.3 UTL burnt area algorithm procedure for Africa 1 83

6.1.4 UTL post-processing procedure for Africa 1 89

6.2 UTL Europe module 89

6.2.1 UTL burnt area algorithm procedure for Europe 90

6.3 UTL Asia module 94

6.4 UTL Africa 2 module 95

6.4.1 UTL burnt area algorithm procedure for Africa 2 95

6.4.2 UTL post-processing procedure for Africa 2 97

7 NRI ALGORITHM MODULE FOR SOUTH-WEST AFRICA 98

7.1 NRI Africa module 98

7.1.1 NRI pre-processing procedure 100

7.1.2 NRI burnt area algorithm procedure for Africa 100

7.1.3 NRI post-processing procedure for Africa 104

8 GVM (STOPPIANA) ALGORITHM MODULE 108

8.1 GVM (Stroppiana) pre-processing module 108

8.2 GVM (Stroppiana) compositing procedure 109

8.3 GVM (Stroppiana) burnt area algorithm procedure 109

8.4 GVM (Stroppiana) algorithm post-processing procedure 113

9 CCRS ALGORITHM MODULE 115

9.1 CCRS pre-processing module 116

9.2 CCRS compositing procedure 117

9.3 CCRS burnt area algorithm procedure ... 117
9.3.1 Implementation of the CCRS algorithm ... 117
9.3.2 Other Input Data ... 118
9.3.3 CCRS algorithm: stage 1 ... 119
9.3.4 CCRS algorithm: stage 2 ... 120
9.3.5 CCRS algorithm: stage 3 ... 120
9.3.6 CCRS algorithm: stage 4 ... 122

10 CNR ALGORITHM MODULE ... 125

10.1 CNR methodology for burnt area detection ... 125
10.1.1 Neural network approach ... 125
10.1.2 Network training ... 126
10.1.3 The AMBRALS model ... 129
10.1.4 Stage 1 of the classification approach ... 129
10.1.5 Stage 2 of the classification approach ... 130
10.2 Implementation of the CNR algorithm ... 131

11 UOE ALGORITHM MODULE ... 132

11.1 UOE pre-processing module ... 134
11.2 UOE compositing procedure ... 135
11.3 UOE burnt area algorithm procedure ... 135
11.4 UOE algorithm post-processing procedure ... 137

12 GVM (BOSCHETTI) ALGORITHM MODULE ... 140

12.1 GVM (Boschetti) module ... 140
12.2 GVM (Boschetti) algorithm implementation ... 142
12.2.1 Inversion of the GVM (Boschetti) model ... 142
12.2.2 GVM (Boschetti) burnt area detection strategy ... 143
12.3 Further developments in the GVM (Boschetti) algorithm ... 145

13 CREATING THE GLOBAL BURNT AREA PRODUCT ... 146

13.1 Extracting the unique region of interest from the buffered window ... 146
13.2 Checking for double burnt area detections in monthly products ... 146
13.3 Mosaicking of processed windows ... 147
13.4 Generating ARC-INFO GRID products ... 148
13.5 Producing the GBA2000 burnt area map ... 149

REFERENCES ... 153

Executive Summary

As part of the Fifth Framework Programme 1999-2002 for *Research, Technological Development and Demonstration* carried out by the Joint Research Centre (JRC), the Global Vegetation Monitoring (GVM) Unit was given the task of implementing a project entitled Global Environment Information System (GEIS). One objective of the project is to provide information on disturbances to the world's vegetation cover. This involves analysis of global Earth Observation (EO) data for inventory of vegetation fires and for mapping of areas burnt. It is in this context that the GVM unit has proposed the Global Burnt Area 2000 (GBA2000) initiative.

The European VEGETATION programme partners have made a satellite image archive available as part of their VEGETATION data for Global Assessment initiative, VEGA 2000, with a financial support of CNES, JRC and VITO as a contribution to the objectives of the Millennium Ecosystem Assessment. It covers the period November 1999 to December 2000. The VEGA 2000 dataset was made available to the whole scientific community through an invitation to submit proposals. Data delivery to JRC from the VEGETATION central processing facility operated by VITO in Belgium was completed by March 2001. JRC then organized data extraction and transfer for the GBA2000 partners.

This report provides detailed information and instructions on technical and computational aspects of GBA2000. In this report, the methods adopted to process the global, daily dataset of SPOT Vegetation S1 images for the year 2000 are presented. The report is divided into sections so that the user can easily comprehend and, if necessary, repeat any of the burnt area algorithms described in this report. Following the main report, annexes provide the text of all of the

programs and tools developed by the author (with colleagues). This report does not present any analysis, assessment or interpretation of the final results. The reader is referred to non-technical reports that accompany this report for this information.

In the first chapter, characteristics of the data are presented and an overview of the approach taken to process the data is given. Chapter two describes the module written to extract data from the data archive for a particular region of interest and time period. Chapter three describes the pre-processing module. Chapter four describes the image compositing module and criteria. Chapters five to twelve describe the technical aspects of implementing operationally the burnt area algorithms developed by the GBA2000 project partners. Chapter thirteen describes the methods used to derive the final global burnt area product. A Technical Annex (provided on a CD-ROM enclosed with this report) contains detailed descriptions of all the programs used. These programs are free software; you can redistribute them and/or modify them under the terms of the GNU General Public License (GNU GPL) as published by the Free Software Foundation, under version 2 of the License. The GNU GPL is given in the Technical Annex document.

Acronyms

AML	ARC Macro Language
ATSR	Along Track Scanning Radiometer
BA	Burnt Area
BSQ	Band Sequential (Data Format)
BRDF	Bidirectional Reflectance Distribution Function
CART	Classification and Regression Trees
CCRS	Canadian Centre for Remote Sensing
CNES	Centre National d'Etude Spatiale (France)
CNR-IREA	Consiglio Nazional delle Ricerche-Istituto per il Rilevamento Elettromagnetico dell'Ambiente (Italy)
CSIRO-EOC	Commonwealth Scientific and Industrial Research Organization-Earth Observation Centre (Australia)
DEM	Digital Elevation Model
DN	Digital Number
ENVI	The Environment for Visualising Images (Image Processing Software)
FDV	First Discriminant Variable
GBA2000	Global Burnt Area 2000
GIS	Geographical Information System
GVM	Global Vegetation Monitoring Unit (JRC, European Commission)
HDF	Hierarchical Data Format
IDL	Interactive Data Language
IFI	International Forest Institute (Russia)
IGBP-DIS	International Geosphere Biosphere Programme - Data Information System
JRC	Joint Research Centre (European Commission)
MEA	Millennium Ecosystem Assessment
MIR	Middle Infrared
MLP	Multi-Layer Perceptron (Neural Network)
NDVI	Normalized Difference Vegetation Index
NDWI	Normalized Difference Water Index
NIR	Near-Infrared
NRI	Natural Resource Institute (UK)
PBA	Potential Burnt Area
SAA	Sun Azimuth Angle
SMAC	Simplified Method for Atmospheric Corrections
SPOT	Système Pour l'Observation de la Terre
SZA	Sun Zenith Angle
SUN OS	SUN Workstation Operating System
SWIR	Short-wave Infrared
SWVI	Short-wave Vegetation Index
TM	Thematic Mapper
TREES	Tropical Ecosystem Environment observation by Satellite (JRC)
UMD	University of Maryland
UTL	Universidade Técnica di Lisboa
VAA	Viewing Azimuth Angle
VZA	Viewing Zenith Angle

1 An overview of data characteristics and image processing

This chapter serves to provide a general overview of the hardware, software and methods utilised in undertaking the processing of SPOT Vegetation S1 images to derive the GBA2000 product. The first section of this chapter discusses the characteristics of the SPOT Vegetation S1 data that will need to be processed, the format of the data and related geographical information. The second section introduces global land cover datasets, how this data was used for the project and how this data was registered with the image data. The third section introduces the main processing modules that were developed. The fourth section introduces the computational methods and systems (hardware and software) used to process the data. The fith section explains how digital elevation models (DEMs) were utilised in the project.

1.1 SPOT Vegetation global S1 data characteristics

Daily, global SPOT Vegetation S1 products for the time period, 1st December 1999 to the 31st December 2000 were stored on a large capacity disk interchange system located physically within the GVM unit of the JRC. For general information on the SPOT Vegetation platform and sensor, please refer to a wealth of literature available in remote sensing journals and on the Internet (see for example, http://www.spot-vegetation.com, http://vegetation.cnes.fr or http://www.spotimage.fr). The rationale behind using the SPOT Vegetation sensor to detect and map burnt areas is not discussed in this technical report (this is covered fully in the individual publications of the GBA2000 partners and referenced in this report). However to remind us, the four spectral bands on the SPOT Vegetation S1 sensor are as follows (also shown are their file naming convention or reference name):

- Band B0 (wavelengths between 0.43 – 0.47 μm in the blue component of the electromagnetic spectrum).
- Band B2 (wavelengths between 0.61 – 0.68 μm in the red component of the electromagnetic spectrum).
- Band B3 (wavelengths between 0.78 – 0.89 μm in the near-infrared (NIR) component of the electromagnetic spectrum).
- Band MIR (wavelengths between 1.58 – 1.75 μm in the short-wave infrared/middle-infrared (SWIR/MIR) component of the electromagnetic spectrum).

SPOT Vegetation S1 products are derived from P products. Therefore, the quality of the S product will be related to the quality of the raw P product. For each pixel, the S1 product

provides an estimate of the ground surface reflectance in each of the four spectral bands. This S product has been derived from the P product after being subject to an atmospheric correction using the SMAC procedure (Rahman and Dedieu, 1994). The geometric viewing conditions for each pixel are also provided. These viewing conditions are comprised of the viewing zenith angle (VZA), the viewing azimuth angle (VAA), the sun zenith angle (SZA) and the sun azimuth angle (SAA) resulting in an additional four bands of data.

Because of the characteristics of the satellite, repeat acquisitions are made in a single day at moderate to high latitudes in both hemispheres. Where this has occurred the pixel chosen to represent the S1 product is that pixel with the highest normalized difference vegetation index (NDVI) value. This leads to significant speckling in some areas and methods to remove or automatically detect these areas of speckle have proved difficult. Therefore, it was a task of the algorithm development team to ensure their burnt area algorithm was sufficiently robust enough to cope with these overlap regions. It was impossible to retrieve the original data so that just one overpass could be used instead of a composite of both overpasses. Because of the same orbital characteristics mentioned previously, there was not complete daily coverage of the equatorial (90% covered each day, the remaining 10% covered the following day).

The maximum off-nadir observation angle of the sensor is 50.5 degrees resulting in a swath width of approximately 2200 km. The imagery were acquired in descending mode at approximately 10:30 local solar time. The S1 product also contained a ‘status’ product that yielded information on the radiometric quality of each pixel and the occurrence of land or water, snow or ice and cloud. However, after discussion with colleagues it became clear that the status map could not be relied upon for providing accurate information to use operationally, for example, to determine the presence of clouds. An example of a global, S1 product is shown in Figure 1. The resampled data is displayed as an RGB image (Red = band MIR, Green = band B3, Blue = band B2). The date of the acquisition is the 30th July 2000.

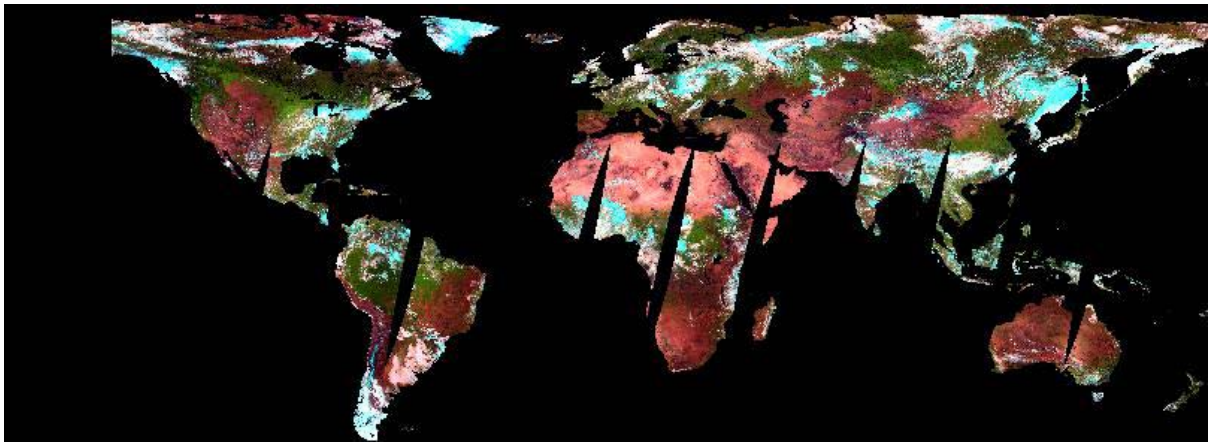


Figure 1: Example of a global SPOT Vegetation S1 product acquired on the 30th July 2000. The data is shown as a RGB image (Red = band MIR, Green = band B3, Blue = band B2) and has been resampled to 2% of its original size.

1.1.1 S1 data format

The daily, global datasets were provided in HDF format, each band having its own HDF file that also included information about the calibration and geometry of the data. The method used to extract the data and relevant calibration information (to ground reflectance or angles) from the HDF file is described in Section 2.1. It became clear that the processing was far easier without having to deal with the HDF format when using a variety of software packages and computer code. Each of the S1 products has been resampled to a Plate-Carree projection (with a pixel spacing of approximately 1 km at the equator) using the WGS84 datum as a reference. All of the daily data were co-registered together prior to delivery to the JRC. The main characteristics of the S1 dataset that was used in this project can be summarised by the following points:

- 4 x spectral bands (B0, B2, B3 and MIR) and 4 x angle bands (VZA, VAA, SZA, SAA).
- Each band contained 40320 columns x 14673 lines of data (global coverage).
- The spectral bands are encoded as unsigned 16-bit (integers). To derive the surface reflectance from the digital number (DN), simply multiply the DN by 0.0005.
- The angle bands are encoded as unsigned 8-bit (byte). To derive the value of the angle from the DN, multiply the DN by 1.5 for azimuth angles or multiply by 0.5 for zenith angles (for both viewing and sun angle values).
- The byte order is IEEE format (big-endian).
- The pixel spacing is 0.0089285714 degrees that approximates to 1 km at the equator.

- The coordinates of the centre of the upper-left pixel are –180.0 degrees E, 75.0 degrees N.

Uncompressed, a daily dataset occupied 6.6 Gbytes of disk space and there are at least 366 days of data to process (the year 2000 was a leap year). Therefore, effective management of hardware and software resources is a key issue in the execution of this project.

1.2 Utilisation of global land cover products

Information on the land cover at a global scale is useful for the following reasons in the context of the GBA2000 project:

- As a tool to provide data masks of water, urban regions and non-vegetated surfaces such as ice masses and deserts.
- As an information layer used to retrieve statistics of burnt area by land cover (vegetation) type.
- As a guide to utilising burnt area algorithms for detecting burnt areas outside their geographical region where they were developed, but over similar land cover (vegetation) conditions.

A number of global products exist that could have been used. Some of them were developed with a specific application or research area in mind, such as the Simple Biosphere 1 and 2 models (Sellers *et al*., 1986; Sellers *et al*., 1996) and some models were too detailed for this application, for example the Global Ecosystems model that has 100 classes (Olson, 1994a; Olson, 1994b). Two models that were considered in this project are the International Geosphere Biosphere Programme (IGBP) land cover classification (Belward, 1996; Belward *et al*., 1999) and the University of Maryland (UMD) global land cover product (DeFries *et al*., 1998; Hansen *et al*., 2000). After lengthy discussion the IGBP product was rejected. This was because using two land cover products in the pre-processing stages would be complicated.

1.2.1 UMD land cover product data access and characteristics

All of the data products mentioned in the previous section (apart from UMD product) can be downloaded online, free of charge, from the US Geological Survey (USGS) EROS Data Centre (http://edcdaac.usgs.gov/glcc/glcc.html). The UMD product can be downloaded online, also free of charge, from the UMD Global Land Cover Facility (http://glcf.umiacs.umd.edu/data.html). This website also provides information on the origin

of the project and the procedure of training the classification model and its validation. The data is available in two projections, either geographic (Plate-Carree) or in Goode's Homolosine. The geographic projection was chosen at a resolution of 1 km at the equator. The characteristics of the UMD land cover dataset are:

- Image size is 43200 columns x 21600 lines of data.
- The data encoding is unsigned 8-bit (byte).
- The projection is geographic based on a sphere of radius 6370997 m.
- The pixel spacing equals 0.00833 degrees.
- The coordinates of the centre of the upper-left pixel are –180.0 degrees E, 90.0 degrees N.

The legend of the UMD land cover dataset is:

Class Value (DN)	Class Name
0	Water
1	Evergreen needleleaf forest
2	Evergreen broadleaf forest
3	Deciduous needleleaf forest
4	Deciduous broadleaf forest
5	Mixed forest
6	Woodland
7	Wooded grassland
8	Closed shrubland
9	Open shrubland
10	Grassland
11	Cropland
12	Bare ground
13	Urban and built-up

Figure 2 shows the global UMD land cover product, having been significantly reduced from its original size.

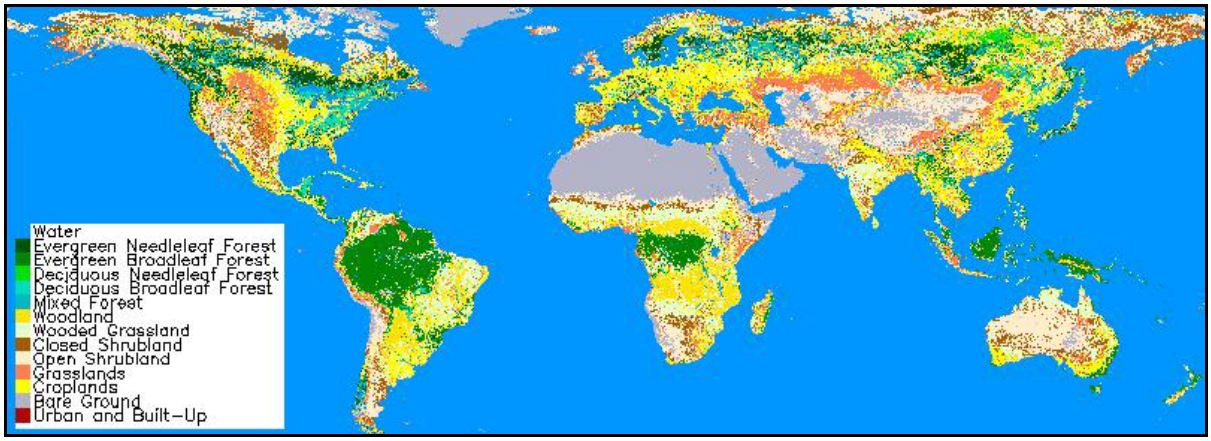


Figure 2: The UMD global land cover product. The image has been resampled to 2% of its original size.

1.2.2 SPOT Vegetation S1 - UMD land cover data co-registration

An initial overlay of the UMD land cover product and the S1 products yielded a significant non-linear offset between the two products that had to be corrected. It was obviously easier to correct the land cover product than all of the S1 products. The commercial software package, ENVI was used to make this correction. In total, 54 tie-points were selected from an S1 product and the UMD land cover product. To perform the correction, a third order polynomial equation was applied to the land cover product with nearest neighbour resampling. The estimated RMS error of the resampling procedure was less than 1 pixel. The resampled land cover product was projected onto the WGS84 datum, with a new pixel spacing of 0.0089285714 degrees and geometry of 40320 columns by 14673 lines (i.e. identical to the SPOT Vegetation S1 data).

1.2.3 Secondary mask products derived from the land cover data

The following secondary products were then derived from the registered UMD land cover dataset:

- A binary mask (0/1), of non-vegetation/vegetation cover. This file would be encoded zero for all classes that were not vegetation (i.e. classes 0 (water), 12 (bare ground) and 13 (urban and built-up)), or otherwise encoded as one.
- A binary mask (0/1), of the vegetation classes that are classified as forest. These are classes 1 (evergreen needleleaf forest), 2 (evergreen broadleaf forest), 3 (deciduous needleleaf forest), 4 (deciduous broadleaf forest) and 5 (mixed forest). This was

considered useful in case two burnt area algorithms needed to be applied over the same area, one that detects burnt area in forests and the other detecting burnt areas in woodlands, shrublands and grasslands.

- A binary mask (0/1), that satisfied the other vegetated land cover classes that are not forest as described in the previous bullet point.

The storage, accessibility and usage of these products in the GBA2000 project are described in Section 3.5.

1.3 A simplified overview of the main processing tasks

Considerable time was given to thinking about, and formulating the sort of processing needs and requirements of the GBA2000 project. For instance, would there be any common pre-processing requirements that satisfied a number of burnt area algorithms. Would the compositing criteria tested successfully with algorithm A over region B work to provide the required composite data for algorithm C over region D. Initially, the work undertaken by the author, was almost to predict what solutions might be required by the GBA2000 partners and to develop and test some programs with the S1 data. From these early tests and correspondence with GBA2000 partners, it became evident that the algorithms being developed were all very different. They each demanded different amounts of processor time, input data formats and post-processing. However, some of the tasks, mainly in the pre-processing module, were similar or could be approximated by using the same tool. It became clear though that each of the burnt area algorithms would need to be treated separately with only the final results being in a comparable format to enable the final product to be produced. It was decided to break the processing up into several main processing tasks, or modules. The first module is called the data extraction module. After extraction, the data now enters the pre-processing module, containing different sub-routines for each algorithm, followed by entry into the burnt area detection modules. Each of the algorithms provided by the GBA2000 partners require different post-processing algorithms. However, the results are comparable and it is the final task to mosaic the results together to create the global burnt area map of the year 2000. In the following chapters and sections, each module is described in detail.

1.4 Computing requirements of GBA2000

The majority of the computational processing for the GBA2000 project was achieved using SUN Workstations. Several workstations were utilised so that the processing could be multi-tasked. The workstations used ranged from SUN Ultra 2 with a single processor to a dual processor Ultra 60. The programs were developed so that they could all be run on a system with 128 Mbytes of RAM with an equivalent amount available as swap space. The memory available in the workstations ranged between 128 Mbytes and 1 Gbyte. The amount of disk space available for the project was a limiting factor on the efficiency and productivity of the processing. Obviously, the bigger the disk (or disks) capacity then a larger amount of data can be processed at the same time. The disk space allocated to GBA2000 was approximately 92 Gbytes allocated in partitions of 17.5, 17.5, 10, 34, 9 and 4 Gbytes. It was considered essential that at least one 30+ Gbytes partition was available as this would significantly decrease the processing time of some of the more demanding burnt area algorithms. The operating systems on the SUN workstations are the standard SUN operating system (SUN OS) versions 5.6 and 5.8. All of the disk partitions were cross-mounted with the processors that were available.

The choice of software used to implement the burnt area algorithms were the c programming language, the Interactive Data Language (IDL) developed by Research Systems Inc. (RSI) and the ARC Macro Language (AML) developed by ESRI. This decision was made because of several reasons; the author of this report is familiar with these programming languages, the software was available on the UNIX network at the GVM unit of the JRC, the archived SPOT Vegetation S1 data was accessed through the UNIX network using OS commands, and c-shell scripts could be developed to automate much of the processing. To enable the processing to be undertaken, c language compilers need to be available on the UNIX network (e.g. cc or gcc). Use was made of the standard commands and software available under SUN OS, such as cat, awk, echo, HDF commands and c-shell scripts. Further information about these commands and others can be found in Annex A or by viewing the manual pages on the operating system. The choice of software that the algorithms were written in also depended on the GBA2000 partner that provided the algorithm. In some cases, the algorithm would be provided as a formula or set of equations that the author would encode. In other cases, code would be provided and, if this code was compatible with the UNIX system at the JRC (such as an algorithm by CCRS that was developed in AML from ARC-INFO (see

http://www.esri.com/software/index.html for details), then the code would be run without any re-writes or changes. It was the job of the author to make the algorithm run automatically. The result is that some of the algorithms are available in the c language, others are in IDL or AML, but they all run on the same system and executed in the same way (i.e. using c-shell scripts). A brief description of the use of c-shell scripts is now given.

1.4.1 Automating the processing using c-shell scripts

The use of c-shells, or any shell environment, to automate processing and tasks is well practiced. By using this software, that is standard on SUN OS, it is possible to keep the processing modules discrete from each other, keep programs from becoming too long and complicated and also allow easy bug fixing if the processing goes wrong. Additional, advantages of c-shell scripts are that tasks can easily be repeated many times and that c programs can be executed automatically as well as IDL programs and AML modules within ARC-INFO. A c-shell script is a simple text file that is interpreted by the operating system as a script with the inclusion of the following information on the first line:

```
#! /bin/csh –f
```

This tells the operating system that the text file is a c-shell script. The text editor used to develop these programs is called, nedit (the author strongly encourages the use of this editor, or another editor of similar quality, for all text editing). A c-shell script is basically a series of commands that are executed in sequence after the previous command has finished. Within the c-shell it is possible to repeat commands within loops, assign variables from command line input values, direct input and output (using, < and >), make simple mathematical calculations and assign text and numerical variables from text files using the commands, awk and sed (type, man awk or man sed for more information). Other manipulations are available. Also, if the code is made generic so that specifics are given at the command line that activates the c-shell script, then they can be run anywhere within the directory structure. Examples of c-shell scripts that illustrate the points made above are given in Annex A.

1.4.2 Location of executable binaries and source code

To avoid replication of command files in multiple directories, it is possible to store all of the binaries (c programs, c-shell scripts, awk programs) within one directory. In the GBA2000 project, necessary files were located in ~tanseke/bin, where ~tanseke, refers to the location of the home directory of user, tanseke (i.e. the author). The directory path, ~tanseke/bin was

then added to the list of those directories looked in when executing commands in the .cshrc file located in the user's home directory (please consult your system administrator or technician for information about this file). The source files of the c programs were kept separately in the directory called ~tanseke/src. A directory called ~tanseke/src/idl contained the IDL programs. The IDL programs are not compiled before use, rather that is done automatically when the program is called. Therefore, before these programs are called, they are copied from this central resource to the working directory (i.e. where the processing is taking place). After the processing is complete, the copy is deleted. The AML programs are located in a central resource, in this case ~tanseke/src/amls. These files can be called and run from this directory without the need to copy the program to the working directory. Also, directories ~tanseke/bin/text and ~tanseke/bin/img have been created that respectively contain important text and image files, required at certain times during the processing.

1.4.3 Computational processing demands geographical area relationships

The large size of the SPOT Vegetation S1 global dataset and the development of regional algorithms ensured that the processing would be undertaken on a regional scale. During early considerations of the processing tasks, it was considered that all of the algorithms could be applied to all of the geographical regions without causing a software or hardware failure on any of the systems available. This was achieved, even if some of the processes would be extremely slow on the older machines. So, based in part on the processing demands of the algorithms (and in some part on the land cover zoning and the physical shape of the land surfaces), the global dataset was divided into regions of interest (sub-windows). The method of extracting the sub-windows is presented in Section 2.5.2. Figure 3 shows these regions of interest and the acronym given to each of these.

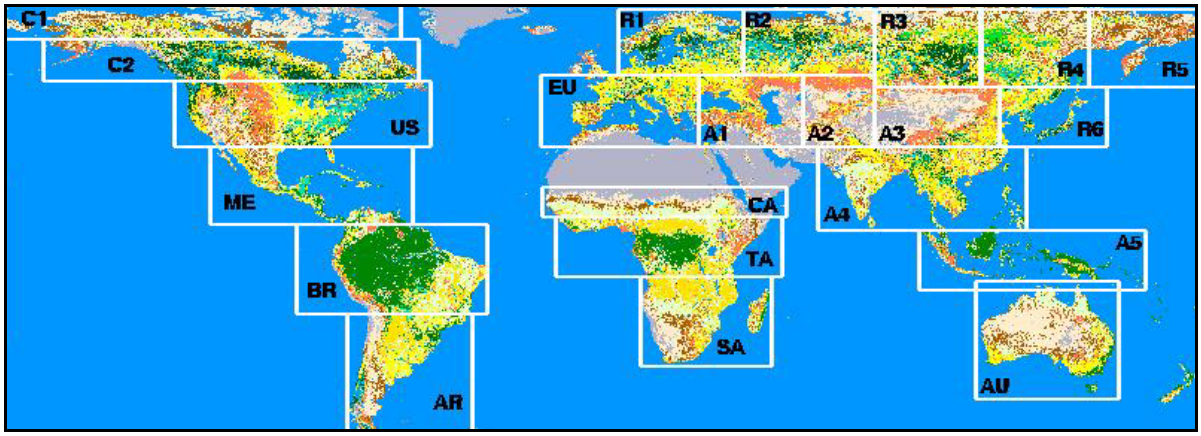


Figure 3: Division of the global dataset into regional sub-windows, imposed because of hardware and software processing limitations. An acronym is given for each sub-window.

Note in Figure 3 that New Zealand, Greenland and sections of the Middle East and the UK were not processed. The low occurrence of vegetation burning in these regions caused by a lack of vegetation or burning regulations made the processing of these areas unnecessary.

1.4.4 Further software requirements

Other software tools that were utilised in the GBA2000 project were the Environment for Visualising Images (ENVI), by Research Systems Inc., a tool that is based on the Interactive Data Language (IDL) and programs of ARC (ARC-VIEW, ARC-INFO and ARC-GRID) by Environmental Systems Research Institute Inc. ENVI is commercially available software that provides tools for the display and manipulation of data. This software was used mainly for the display and viewing of image files, overlay of vector information, creation of maps for printing and projection of Plate-Carree data to equal area projections. Using ENVI to display images requires that a header file with the same file name (but with the extension .hdr) is available (if not one is created). A feature of many of the GBA2000 programs is the automatic creation of the ENVI header of the binary image data, so that these images can be displayed quickly in ENVI. The ENVI header file, in its basic form, is a simple text file that provides geometrical and geographical information. The header file provides information on any type of binary image, for example the global land cover products, which are effectively classifications with a legend. The structure of the ENVI header file is described in Annex A and two examples are given. Even if you do not utilise ENVI, the header file still provides useful information about the image file. The versions of ENVI and IDL used were 3.4 and 5.4.1 respectively.

The programs available within ARC were used to create ARC Grids from the binary image data (burnt areas and land cover products) and projection to equal area (ARC-INFO or ARC-GRID command). The AML was used for a burnt area algorithm, described in Chapter 9, where complete description of the commands used to derive the burnt are products are presented. The versions of ARC and ARC-VIEW used were 8.0.1 and 3.2 respectively.

1.5 Utilisation of digital elevation model (DEM) data

During the processing of the GBA2000 SPOT Vegetation S1 dataset, in the regions of the Himalayas and the Andes mountain ranges large regions of possible burnt areas were detected by the algorithms used in these regions. After closer inspections of the results were made, it was believed that these areas were being mistaken as being burnt because these areas were lying in the sun's shadow during certain times of the year and this would affect the pixel's spectral reflectance. The drop in reflectance would, in some cases, be falsely interpreted as being a burnt area. The regions identified as needing attention were the Andes mountain range of South America (windows ME, BR and AR in Figure 3) and the Himalayas mountain range (window A4 in Figure 3). It was necessary to determine those pixels that could be affected by the sun shadow at different months of the year. To achieve this, a DEM was needed.

After a search of the Internet of available global DEM's, the Global Land One-km Base Elevation (GLOBE) product was used (Hastings and Dunbar, 1998; see also http://www.ngdc.noaa.gov/seg/topo/globe.shtml). The DEM data can be downloaded by geographical region from the Internet. In addition, a layer corresponding to the source of the DEM is available. The data format is signed 16 bit and the byte order is INTEL (little-endian). The individual tiles for two regions were joined together using IDL ENVI software. These two regions were Central and South America and the Indian subcontinent, continental south-east Asia and insular south-east Asia. The projection of the GLOBE data is geographic (degrees) and the pixel size is 0.008333333 degrees. Because of the small difference in pixel size between the SPOT Vegetation S1 products and the DEM product, the DEM product was resampled to the pixel size of the image data. A visual confirmation of a good match was made. Any form of accurate co-registration was impossible given the types of data being used. For each of the regions of interest, where it was thought that sun shadow might lead to false detections, a subset of the DEM was extracted. These regions were ME (Mexico), BR

(Brazil), AR (Argentina) and A4 (India, China and south-east Asia). The use of a DEM was not considered necessary for window A5 (insular south-east Asia).

1.5.1 Determining sun shadow affected pixels

The method used to determine pixels that are contaminated by sun shadow is given by Colby (1991). The cosine of the solar incidence angle is determined by the following equation:

$$\cos_i = \cos \theta_s \cos \theta_n + \sin \theta_s \sin \theta_n \cos (\phi_s - \phi_n)$$

where, $\cos_i$ is the cosine of the solar incidence angle, θ_s is the sun zenith angle, θ_n is the terrain slope, ϕ_s is the sun azimuth angle and ϕ_n is the terrain aspect.

To derive the terrain slope and aspect, IDL ENVI software was used (note that this task can be performed by most commercial image analysis software). The slope and aspect products were derived over a 3x3 window and assuming a pixel size on the ground of 1km. Given the time constraints of the project, there was insufficient time to undertake a sensitivity analysis of the optimum threshold to apply to the cosine of the solar incidence angle to indicate that a pixel is affected by sun shadow. After some brief tests were conducted in the Andes of South America a value of 0.5 for the cosine of the solar incidence angle was chosen. This means that if the incidence angle of the solar radiation on the surface representing each pixel was below sixty degrees then this pixel would be indicated as being contaminated by sun shadow. The sun zenith and azimuth angle information is derived from the SPOT Vegetation S1 products on a daily basis.

A c program was written that made the calculations shown in the equation above. To enable the threshold value to be changed by the user, this parameter was made an input variable at the command line. A c-shell script was written to automatically produce sun shadow contamination masks for any time period.

It was decided to use these masks in the post-processing stages of the GBA2000 project. This was mainly because, the user needed to determine whether or not the application of these masks to the final burnt area maps was actually necessary. Therefore, this meant that sun shadow masks that indicated all of those pixels that were contaminated for the whole of the month under consideration (and not just the each day). For example, pixels not contaminated at the beginning of a month may be affected by sun shadow at the end of the month. If a

monthly burnt area product is presented then all of those pixels possibly contaminated by sun shadow during the whole of that month must be removed. Finally, for some of the algorithms, tests are made between images (or composite images) acquired over two months. In this case, all contaminated pixels in both months must be considered.

1.5.2 Sun shadow masking procedure

The procedure adopted by the GBA2000 project was to determine the contaminated pixels for each daily image set over a time period of one month using a sun incidence angle cosine value of 0.5. Because of the assumptions made of the pixel size and accuracy of the DEM product, the sun shadow masks were eroded by one pixel in all directions. Although this resulted in a loss of data, it was considered a necessary step after observations of the results in the Andes test area. The daily masks for each month were then multiplied together to obtain a single mask for each month indicating all contaminated pixels. Where tests were made over a two month period to determine burnt areas, these monthly products for these two months would be multiplied together to derive a bi-monthly mask. The DEM derived products (slope and aspect images) are located in a directory called dem_products located in the working directory of each region of interest (e.g. ./BR/dem_products).

The c-shell script gba_sun_shadow_mask.csh has been written to automate the production of the monthly sun shadow masks. The syntax of the script is:

```
#########################################################
GBA2000 C shell: gba_sun_shadow_mask.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_sun_shadow_mask.csh
<acronym>
<apply mask to data to which level (0 or none, vza, mir, cloud, cl_comp, shadow, burn, basic or all)>
<cosine of sun incidence angle threshold (e.g. 0.5)>"
<dilate sun incidence angle mask by 1 pixel (0 = no, 1 = yes)>"
<delete mask (1,0) files (1 = yes, 0 = no)>"
```

where, the acronym is the two letter window code, the apply mask parameter indicates to the program the level of pre-processing that the user requires to be composited (the codes point the program to the correct input files that have been produced by the pre-processing algorithms presented in Chapter 3 and a zero here indicates that original S1 data is to be processed), the threshold value of the cosine of the solar incidence angle is stated, a flag value stated to indicate if the user would like to erode (or dilate) the sun shadow masks in each direction by one pixel and the final parameter indicates that the user would like to delete

the daily masks and also a file containing values of the cosine of the solar incidence angle. Further details of this program are given in Annex A.

The script calls a c program called sun_shadow_mask.c. The inputs to this program are the following:

sun_shadow_mask
terrain aspect image (float)
terrain slope image (float)
sun azimuth angle image (byte)
sun zenith angle image (byte)
output cosine of solar incidence angle image (float)
output sun shadow mask image (byte)
$pixels $lines
cosine of solar incidence angle threshold

A full description of this program is given in Annex A.

After a sun shadow contamination mask has been produced for each day, a monthly product is produced using the generic c program apply_mask.c (Annex A). This file is written to a directory named sun_shadow_masks located in the working directory of the region of interest. Where GBA2000 algorithms utilise these sun shadow masks then this is described in the post-processing section of the individual algorithm descriptions contained within this report.

2 Data extraction from tape archive

The daily, global S1 image dataset acquired between 1st December 1999 and 31st December 2000 is stored on tape archive within the GVM Unit. The data on the tape archive can only be accessed by a limited number of users at the same time, and two users cannot access the same tape at the same time. The amount of data available is huge and, for obvious reasons, cannot all be accessed for processing at the same time. Therefore, programs were developed so that specific regions of the globe and specific dates could be retrieved from the archive and presented in an orderly manner automatically. This chapter first describes the format of the archived SPOT Vegetation data, then outlines the programs required to extract a particular region of interest for a particular time period in sections two and three. The extraction of land cover information is covered in the fourth section. The fifth section describes a special case that is considered when extracting the image data for some regions of the globe. The final section lists the daily datasets that have problems associated with them (i.e. missing days, different file names, missing data etc.).

2.1 Data format of original SPOT Vegetation S1 archived products

The original S1 products are located in the remote directory:

/gvmarchive/vgtglday$month$year/$year$month$day

where, the variables pre-fixed by a dollar sign ($) indicates the year (e.g. 2000), month (e.g. 11) and day (e.g. 23) of acquisition.

Inside this directory, the following files are available (example is shown for the 15th June 2000):

./0001_b0.zip
./0001_b2.zip
./0001_b3.zip
./0001_log.zip
./0001_mir.zip
./0001_ndv.zip
./0001_ql.zip
./0001_rig.zip
./0001_saa.zip
./0001_sm.zip
./0001_sza.zip
./0001_tg.zip
./0001_vaa.zip
./0001_vza.zip

This file naming convention is the same for each of the daily datasets, hence, it is important when retrieving the data to keep track of which date the file represents. The author refers the reader to the SPOT Vegetation data manual for a more complete description of the content of the files stated above (available at http://vegetation.cnes.fr). The files that are important to the GBA2000 project are those with the extensions b0, b2, b3, mir, saa, sza, vaa, vza and log. The first four files contain the spectral information, the next four files contain the angular information and the final file (log) contains geometric, cartographic and projection information.

To save disk space each file has been compressed. When decompressed, the spectral and angular images have the extension .HDF. The data is in Hierarchical Data Format (HDF) (see http://hdf.ncsa.uiuc.edu/ for more details). This data format is very useful when multiple files and meta-data information need to be structured in such a way that they are contained within the same file, such as with Landsat TM data. However, in this case there is only a single band of information contained within each HDF file and the geometry (columns and lines) of this single file is always known. In addition, most of the meta-data contained within the HDF file is not required by the GBA2000 project. Furthermore, image manipulation with HDF files makes processing far more complicated. Given all of these factors, it was decided that, when extracting the region of interest from the data archive, firstly the image files would renamed to something that could easily be associated with the region of interest (by using an acronym) and the date of image acquisition (giving the file a date code), second, the image data would be extracted from the HDF file and written as simple BSQ (band sequential) file with no internal header information, and third, writing the files to a directory structure that enables the user to easily understand and locate data products from their location within the directory.

The structure of the directory to which the extracted datasets were written was made up of a root directory that was named after a two-letter acronym given to each of the geographical regions (e.g. AU refers to Australia). All of the geographical regions analysed by the GBA2000 project are shown in Figure 3 along with their acronyms. In the case for Russia, the country was divided into several sections, numbered consecutively. As the files were extracted for each date requested, a sub-directory would be created (inside the root directory), of the form $year$month$day (e.g. 20000615). Inside each of the sub-directories would be written the image data, angular data and calibration information, for that day and for that

particular geographical region identified by the acronym preceding the date, as shown below (example of Australia is given):

```
./AU/20000614/…
./AU/20000615/AU_20000615_b0
./AU/20000615/AU_20000615_b2
./AU/20000615/AU_20000615_b3
./AU/20000615/AU_20000615_mir
./AU/20000615/AU_20000615_saa
./AU/20000615/AU_20000615_sza
./AU/20000615/AU_20000615_vaa
./AU/20000615/AU_20000615_vza
./AU/20000615/20000615_log.txt
./AU/20000615/20000615_calib_par.txt
./AU/20000616/…
```

where, the log and calibration parameter files are in ASCII text format and provide metadata information for projecting and calibrating the spectral data.

This data structure is understood by programs developed by the author for processing the SPOT Vegetation dataset into a burnt area product.

2.2 Controlling the time period of analysis: input_dates.txt

The text file named input_dates.txt is absolutely critical for controlling the GBA2000 processing. It is evident that for many regions of the globe the occurrence of vegetation fires is during a specific time period. Therefore, it is pointless trying to retrieve data spanning the whole of year 2000, if there are no burnt areas to detect. Information on the timing of burning in different regions of the globe was obtained from discussion with local experts, reading literature on the subject and examining images from a range of dates throughout the year. It was also a waste of computing resources to process data that do not contain any burnt areas. Literature on the timing of vegetation fires has been well published (Dwyer *et al*., 1998, 1999; Barbosa *et al*., 1999). To control the range of dates that the user needs to process, a text file is used. An example of the text file is:

```
# Input file listing dates for processing of the GBA product
# Input year, month and days you want to process

years = 2000
months = 07
# days = 21

# For full image processing use these value
# years = 1999 2000
# months = 01 02 03 04 05 06 07 08 09 10 11 12
days = 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
```

The file is named input_dates.txt and a copy should always be available in a central resource (e.g. ~<user>/bin/text/). Copy the file to the current working directory (e.g. ./AU) and edit the file using a suitable text editor (e.g. nedit). All of those lines that begin with a hash symbol (#) are ignored. The details are repeated so that the lower sets of values are given to represent the whole dataset, i.e. all days in all months in all years. The upper set of values can be altered and modified according to the user's requirements. By copying and pasting values from the lower set of values into the upper set of values mistakes cannot be made. In the example shown, a program that needs temporal information, when activated at the command line will read in the file, input_dates.txt and see that it needs to process each day in the month of July (month 07) for the year 2000 dataset. Programs will test for the presence of this file within the working directory and may fail if this file is not located.

2.3 Extracting SPOT Vegetation S1 data from tape archive

A c-shell script, that calls a number of c programs, is the method used to extract the data, for a specified region, from the archive. The script then renames the files and strips the HDF structure from the original SPOT Vegetation S1 products. The script also extracts for the same geographical region the UMD land cover products and masks that are described in Section 2.4. To retrieve a specific geographical region, the location of the region (or window) must be known in terms of pixels offset in columns and lines relative to the start position (0, 0) of the upper-left pixel. The coordinate (0, 0) is chosen as this is better represented in c-shell and c programming. This can be calculated in several ways either by looking at the global product and determining the map pixel offset in columns and lines or using the geometry of the file, given the geographical coordinates of the window and the pixel spacing in degrees. The offset, measured in pixels, of the upper left corner of the region of interest, compared to 0, 0 of the global product. For example, the Australian dataset is offset 32702 columns (also known as pixels) and 9519 lines relative to the global product. In the extraction program, these offsets are termed x_start and y_start respectively. The user needs to know the size of the geographical dataset, also in terms of columns (pixels) and lines. Using Australia again as an example, the size of the window is 4817 columns (pixels) by 3921 lines. In the extraction program, these parameters are known as x_size and y_size respectively. The program initially checks a list of pre-defined regions of the globe. This list is in the form of a text file that is located in a central resource. The list of regions is called gba_country_list.txt and is located in the directory, ~<user>/bin/text. An extract of that list is now shown.

```
# List of GBA2000 Regional Sub-windows
# Legend is:
# country_name acronym x_start y_start x_size y_size

# Regional Sub-Windows
australia AU 32722 9539 4777 3881
canada1 C1 20 20 13400 1380
canada2 C2 1400 1400 12600 1400
usa US 5800 2800 8600 2200
mexico ME 7000 5000 6800 2600
brazil BR 9900 7600 6400 3000
argentina AR 11600 10600 4200 4000
c_africa CA 18150 6350 8200 1000
t_africa TA 18600 7350 7600 2000
s_africa SA 21450 9350 4400 3000
europe EU 18100 2600 5400 2400
russia1 R1 20700 400 4200 2200
russia2 R2 24900 400 4400 2200
russia3 R3 29300 400 3600 2600
russia4 R4 32900 400 3600 2600
russia5 R5 36500 400 3800 2600
russia6 R6 33500 3000 3600 2000
asia1 A1 23500 2600 3400 2400
asia2 A2 26900 2600 2400 2400
asia3 A3 29300 3000 4200 2000
asia4 A4 27400 5000 7000 2800
asia5 A5 30800 7800 7600 2000
```

The first column of data is matched with the input variable given, the second column is the acronym given to the sub-window and the following four columns indicate the location and the size of the sub-window relative to the global product. This file can be added to and edited to suit the requirements of the user

The c-shell script gba_read_vgt_data.csh is available to perform all of the tasks outlined above in an automated way. To extract a dataset, essentially all the user has to do is edit the file, input_dates.txt for the required time period, and specify either one of the listed GBA2000 processing windows (e.g. the first section of Canada, C1 or Australia, AU) or specify the input parameters (x_start, y_start, x_size and y_size) and provide an acronym for the geographical region (e.g. IT for Italy) and run the program. By specifying more detail at the command line when running the program, it is possible to extract land cover information and/or retrieve the geographical region, buffered by twenty pixels to provide better cloud shadow detection results. These add-in's to the main program are discussed in Section 2.5.

If the user types gba_read_vgt_data.csh at the command line instructions appear on how to use the program. A description of the procedures followed by the script is now given, a more complete description and the code of the script is given in Annex B.

```
#######################################################
GBA2000 C shell: gba_read_vgt_data.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_read_vgt_data.csh
<country>
<extract land cover (1 = yes, 0 = no)>
<apply pixel buffer (1 = yes, 0 = no)>
<acronym> <x_start> <y_start> <x_pixels> <y_lines>
Notes:
Copy and edit the file '~tanseke/bin/text/input_dates.txt' to the current directory
Edit the file '~tanseke/bin/text/gba_country_list.txt.txt'
Select from:
globe = global product (GL)
GBA2000 continental datasets:
australia = continental Australia (AU)
canada1 = Canadian far northern latitudes (C1)
canada2 = Canadian northern latitudes (C2)
usa = United States of America (US)
mexico = Mexico and Central America (ME)
brazil = Brazilian Amazon and northern S. America (BR)
argentina = southern S. America (AR)
c_africa = central Africa (CA)
t_africa = tropical Africa (TA)
s_africa = southern Africa (SA)
russia1 = European Russia (R1)
russia2 = west Asian Russia (R2)
russia3 = central Asian Russia (R3)
russia4 = eastern Asian Russia (R4)
russia5 = far east Asian Russia (R5)
russia6 = south-east Russia and Japan (R6)
europe =  southern and central Europe (EU)
asia1 =  Black Sea, Turkey and Middle East (A1)
asia2 =  central Asia (A2)
asia3 =  northern China and Mongolia (A3)
asia4 =  Indian sub-continent and mainland southern Asia (A4)
asia5 =  Insular south-east Asia (A5)
Flag extract land cover = 1 to retrieve UMD window
Flag apply pixel buffer = 1 to make buffer (value set in this program)
Otherwise enter 'other' for country, acronym and coordinates of the image sample
```

Initially, we will ignore any extraction of land cover images. So, entering the command:

gba_read_vgt_data.csh russia3 0 1

This will tell the program that you want to extract the window russia3 with no corresponding land cover information but with a buffer of twenty pixels. Inside the file gba_country_list.txt parameters for the acronym, global offset and size of the window of region russia3 are listed. In another situation, the user could enter the command:

gba_read_vgt_data.csh other 0 0 MD 24980 9730 850 1560

This will tell the program that you want to extract a window that is not listed in the country list text file, by defining the input parameter as, other. The user also does not want to extract the corresponding land cover information (first flag set to zero) or apply a buffer around the

window (second flag set to zero). The user has defined the acronym MD for this region of interest. The offset in pixels from the upper-left corner of the global product is 24980 columns (pixels) and 9730 lines. The size of the region of interest is 850 columns (pixels) by 1560 lines. This area corresponds to the country of Madagascar.

On execution of the command for any region of interest the procedures that are activated are approximately the same, and therefore can be described in general terms. An example is given for extracting data files covering the Australian (AU) window. The first step is to test for the presence or absence of files. These include the file input_dates.txt and the remote directory on the tape archive containing the image data for the date that is being extracted. If the directory does not exist for that day then the program moves onto the next date in the input_dates.txt file. In this example, the data for the date of interest is available (15th June 2000) and a directory $year$month$day is created in the working directory (e.g. ./20000615). The file 0001_log.zip is copied to a temporary directory (./temp_hdf) and uncompressed using the command unzip. The file is renamed and placed in the date directory recently created (e.g. ./20000615/20000615_log.txt). An example of this log file can be found in Annex B. A file called ./20000615/20000615_calib_par.txt is then created using the SUN OS command echo. This file will contain the calibration factors for the spectral bands and angular bands for this particular date. This information is contained within the metadata of the HDF files.

The program now repeats the following procedure for the four spectral bands (of type integer) and the four angle bands (of type byte). The procedure is described in detail for the MIR band. The data, in a compressed format, is copied from the tape archive to a temporary directory. The file is then decompressed and renamed within the temporary directory with the following form, for example ./temp_hdf/20000615_mir.hdf. The HDF file needs to be analysed to locate the position within the HDF file of the data to be extracted in BSQ format and also the calibration information. The program calls a small c-shell script gba_read_hdf.csh to interpret the structure of the HDF file. This c-shell script is activated at the command line with the command:

gba_read_hdf.csh
temp_hdf/$year$month$day_$image.hdf
>> temp_hdf/$image.hdf.txt

where, the variable image in this case refers to the band being processed (e.g. MIR) and the >> indicates that the output from the c-shell script should be redirected into a text file, in this example to mir.hdf.txt.

When the c-shell script gba_read_hdf.csh is activated it calls a special command that is able to interpret the HDF file. These commands are usually available on the SUN OS (or can be downloaded from HDF websites, see http://hdf.ncsa.uiuc.edu/). The command used for this purpose is hdfed (i.e. HDF editor). By associating input parameters with this command it is possible to print to a text file, information on the location (byte offset) and size (number of bytes) of files within the HDF file. More information and the code for this c-shell script are given in Annex B. An example of the output text file created is:

(1) Version Descriptor : (Tag 30)
Ref: 1, Offset: 202, Length: 92 (bytes)
(2) Vdata Storage : (Tag 1963)
Ref: 2, Offset: 294, Length: 120 (bytes)
(3) Vdata : (Tag 1962)
Ref: 2, Offset: 414, Length: 85 (bytes)
(4) Vdata Storage : (Tag 1963)
Ref: 3, Offset: **499**, Length: **240** (bytes)
(5) Vdata : (Tag 1962)
Ref: 3, Offset: 739, Length: 90 (bytes)
(6) Scientific Data : (Tag 702)
Ref: 5, Offset: **829**, Length: **1183230720** (bytes)
(7) Vdata Storage : (Tag 1963)
Ref: 6, Offset: 1183231549, Length: 4 (bytes)
(8) Vdata : (Tag 1962)
Ref: 6, Offset: 1183231553, Length: 60 (bytes)
(9) Vdata Storage : (Tag 1963)
Ref: 7, Offset: 1183231613, Length: 58692 (bytes)
(10) Vdata : (Tag 1962)
Ref: 7, Offset: 1183290305, Length: 60 (bytes)
(11) Vgroup : (Tag 1965)
Ref: 8, Offset: 1183290365, Length: 37 (bytes)
(12) Vdata Storage : (Tag 1963)
Ref: 9, Offset: 1183290402, Length: 4 (bytes)
(13) Vdata : (Tag 1962)
Ref: 9, Offset: 1183290406, Length: 60 (bytes)
(14) Vdata Storage : (Tag 1963)
Ref: 10, Offset: 1183290466, Length: 161280 (bytes)
(15) Vdata : (Tag 1962)
Ref: 10, Offset: 1183451746, Length: 60 (bytes)
(16) Vgroup : (Tag 1965)
Ref: 11, Offset: 1183451806, Length: 37 (bytes)
(17) Vdata Storage : (Tag 1963)
Ref: 12, Offset: 1183452041, Length: 4 (bytes)
(18) Vdata : (Tag 1962)
Ref: 12, Offset: 1183452045, Length: 61 (bytes)
(19) Number type : (Tag 106)
Ref: 13, Offset: 1183452106, Length: 4 (bytes)
(20) SciData dimension record : (Tag 701)
Ref: 13, Offset: 1183452110, Length: 22 (bytes)
*(21) Numeric Data Group : (Tag 720)
Ref: 4, Offset: 1183452132, Length: 16 (bytes)
(22) Vgroup : (Tag 1965)
Ref: 14, Offset: 1183452148, Length: 59 (bytes)
(23) Vgroup : (Tag 1965)
Ref: 15, Offset: 1183452207, Length: 96 (bytes)

From the geometry of the global dataset and knowing the number of bytes per pixel, we know what the total number of bytes that the image occupies. In the case of the spectral data this is, 40320 multiplied by 14673 and multiplied by 2 = 1183230720 bytes and half of this value for the angular data. If we look at the example shown above we can see this value within the file (shown in bold type). On the same line as this value is another value indicating the offset (in bytes) to the image. To summarise, from this text file, we know the size and characteristics of the BSQ image we would like to extract, and the offset to the beginning of the data within the HDF file. In addition, we have already established the pixel offset (in columns and lines) from the global dataset to the regional dataset that we want to extract. Therefore, we can use the generic c program snip.c (Annex A) to extract from the HDF file the exact region of interest without firstly extracting the global BSQ file and then extracting the smaller region of interest. The program checks for the value of 1183230720 in the 6th column (in this case, a column is separated by a white space) and then reads to a variable the value in the 4th column (the offset which in this example is 829 bytes). In the case of the angular data the value looked for in the 6th column is 591615360.

For data calibration, details provided by documentation from SPOT IMAGE, indicated that the information required is located in a section with length 240 bytes as shown above in bold type. Again the offset to this section of the file was written to a variable. This section of the data was extracted first using the generic c program snip.c:

```
snip temp_hdf/$year$month$day_$image.hdf
temp_hdf/cal_values_$image.txt
100 100 0 0 240 1 $cal_offset 1
```

where, the variable cal_offset is the offset in bytes to the calibration information section of the HDF file.

All 240 bytes of information are extracted from the HDF file and written to a text file. This file is then analysed and the calibration parameters extracted using commands based on awk (see the manual pages on the SUN OS for a description of awk). These parameters are then written to the file called $year$month$day_calib_par.txt. After all the bands have been processed resembles this example from the 15th June 2000:

```
Calibration Parameters for Spectral Bands and Sun/Viewing Angles (date=20000615)
--------------------------------------------------------------------------------
############################################################
Formula:
real_val (reflectance or angle in degrees) = a * DN + b
############################################################
b0 calibration coefficient 'a' = 0.0005
b0 calibration coefficient 'b' = 0.0
b2 calibration coefficient 'a' = 0.0005
b2 calibration coefficient 'b' = 0.0
b3 calibration coefficient 'a' = 0.0005
b3 calibration coefficient 'b' = 0.0
mir calibration coefficient 'a' = 0.0005
mir calibration coefficient 'b' = 0.0
saa calibration coefficient 'a' = 1.5
saa calibration coefficient 'b' = 0.0
sza calibration coefficient 'a' = 0.5
sza calibration coefficient 'b' = 0.0
vaa calibration coefficient 'a' = 1.5
vaa calibration coefficient 'b' = 0.0
vza calibration coefficient 'a' = 0.5
vza calibration coefficient 'b' = 0.0
```

Even though the calibration constants remained the same for S1 products this exercise is useful to ensure that no alternative values exist for the whole dataset.

To extract the specified region of interest, the program again uses the c program snip.c:

```
snip temp_hdf/$year$month$day_$image.hdf
$year$month$day/$acronym_$year$month$day_$image
$global_pixels $global_lines $x_start $y_start $x_pixels $y_lines
$offset $bytes_per_pixel
```

where, the values of x_start, y_start, x_size and y_size come from the parameters of the region of interest outlined previously. The variable offset is the value of the number of bytes from the beginning of the HDF file to the location when the data begins and the variable, bytes_per_pixel is given a value of two for the spectral data (integers) and one for the angle data (bytes).

The output file is written to the date directory created previously and is called ./20000615/AU_20000615_mir. A header file is then created, the ENVI software can immediately interpret that. The format of this header file is described in Section 1.4.4 and in Annex A. To enable the projection information to be written to the header file of the region extracted, new geographic coordinates of the upper left pixel need to be calculated. This is done using the generic c program new_coordinates.c (Annex A). The command to calculate the new longitude and then the new latitude of the upper left corner of the region of interest is:

new_coordinates $top_left_lon 0 $x_start $pixel_size > temp_new_coordinate_lon
new_coordinates $top_left_lat 1 $y_start $pixel_size > temp_new_coordinate_lat

where, the top_left_lon is the longitude value of the global dataset (i.e. –180.0), the zero (0) flag indicates that the calculation is being made for longitude (as opposed to one (1) for latitude in the second command), the value of x_start indicates the offset in columns (in units of pixels) to the region of interest and the pixel_size is the pixel spacing of both products (in decimal degrees). The same is true for calculating the new latitude value.
The new coordinates are written to the header file. The procedure is then repeated for four spectral bands and four angle bands.

The final stage of the extraction program (after the extraction of all dates listed in the control file input_dates.txt) is to write out to the working directory a text file containing important information that is referred to in later stages of the GBA2000 project. This file is called $acronym_file_info.txt, where the acronym indicates the region that is being processed (e.g. ./AU_file_info.txt for Australia). In later stages of processing, the presence of this file is tested for, and many programs will not run without this file being available. Therefore, it is important that this file is created either using this c-shell script or created using your own text editor. An example of this text file is now shown for the Australian sub-window (./AU_file_info.txt).

```
acronym = AU
global_x_offset = 32702
global_y_offset = 9519
samples = 4817
lines = 3921
start_lon = 111.982142
start_lat = -9.991071
pixel_size = 0.0089285714
```

The file contains information on the relative position of the Australian sub-window within the global dataset, the number of samples (equal to columns and pixels) and lines of the sub-window, the geographic coordinates of the upper left pixel and the pixel spacing. The program is now finished and returns to the command line.

2.4 Simultaneous extraction of registered UMD land cover information

By selecting the necessary flag when implementing the c-shell script gba_read_vgt_data.csh it is possible to extract registered UMD land cover and mask products for the specified region of interest. For example, enter the command:

gba_read_vgt_data.csh australia 1 0

The program will extract the image data for all the required dates specified within the file, input_dates.txt and then extract the associated land cover products for the Australian sub-window. The land cover dataset and masks were compressed, using gzip and stored in a central resource that could be accessed from any cross-mounted disk. The first step of extracting the land cover information is to copy the four products (one land cover classification and three masks) from this central resource to the working directory. For each of these products the file is decompressed and then, using the generic c program snip.c, the region of interest extracted from the global dataset and renamed in accordance with the naming conventions previously adopted. An example of the command used is:

snip gl_umd_3order_fine
$acronym_umd_lcc
$global_pixels $global_lines
$x_start $y_start
$x_pixels $y_lines 0 1

where, the output file in named acronym_umd_lcc and is placed in the working directory.

The same process is applied to the three mask files (vegetated surfaces, forested surfaces and non-forested but vegetated surfaces). The header files from the global datasets are also copied into the working directory. The values that have changed within these files are the geometry and projection information. These are re-written using commands based on sed that are standard on the SUN OS. In cases where the user requires only the land cover information, without the need to extract spectral data, just edit the file input_dates.txt so that it indicates that you want to extract data for the date 31st June 2000 (i.e. 20000631). Of course this date does not exist and therefore the extraction of any spectral data will not occur and the processing will move directly to the extraction of land cover products. If the user extracted three dates with accompanying land cover information the directory structure would resemble this example from Australia:

./20000615/20000615_calib_par.txt
./20000615/20000615_log.txt
./20000615/AU_20000615_b0
./20000615/AU_20000615_b0.hdr
…
./20000615/AU_20000615_vza.hdr
./20000616/…
AU_file_info.txt, AU_umd_lcc, AU_umd_lcc.hdr, AU_umd_lcc.mask, AU_umd_lcc.mask.hdr
AU_umd_lcc_forest.mask, AU_umd_lcc_forest.mask.hdr, AU_umd_lcc_non_forest.mask
AU_umd_lcc_non_forest.mask.hdr
input_dates.txt

The dates extracted were the 15th and 16th of June 2000. The land cover information is in the same geometry as the extracted data files.

2.5 Special consideration for specific GBA2000 data extractions

Special consideration was given when extracting SPOT Vegetation S1 data over certain regions of interest. A problem can occur when division of the global datasets is made over land. As presented in the next chapter, the detection of cloud shadow depends on initially the detection of clouds and then the projection of the cloud shadow pixels from the cloud pixel given viewing and sun geometry information. It is theoretically possible and quite evident in some images that at the border of two sub-windows the cloud pixel is located in one of the images and pixels contaminated with cloud shadow present in the other image. If the two sub-windows were processed independently, as is the case within the project, then the pixels contaminated with cloud shadows would not be masked out because the cloud pixel does not exist in the image. The problem only occurs at the edges of the sub-windows. The solution to this problem was to create a buffer around each sub-window and after the pre-processing equations have been applied or after the final results have been produced, remove this buffer. The width of this buffer is set within the c-shell script gba_read_vgt_data.csh. The value selected for operational use is twenty pixels. This value was selected after a visual inspection of contaminated areas and geometrical consideration of the maximum cloud height. To extract a sub-window with a buffer the respective flag must be set to one in the c-shell script gba_read_vgt_data.csh as shown here:

gba_read_vgt_data.csh australia 1 1

where, the first one indicates that the land cover information for Australia should also be extracted and the second one indicates that a twenty-pixel buffer will surround the sub-window.

The output will be similar to the non-buffered windows, except for the fact that the files will be bigger. If you extract the land cover product from the global dataset, this will also contain the same pixel buffer as the satellite data. Two text files containing information on the extracted data will be written instead of one. One contains information about the buffered data and the other about the non-buffered data. Again there is a difference in the naming convention that enables the files to be distinguishable. For those sub-windows covering the

Russia and Asia windows (A1, A2 and A3) the extraction of a buffered window is done automatically.

2.6 Missing or problematic data on the tape archive

The global dataset for the year 2000 (and parts of 1999) is not totally complete. Here is the list of those data that are missing, corrupted or contain erroneous pixels values.

- For date 29th November 1999 (19991129), the MIR and all of the angle bands are absent from the data archive. This date is also not considered in any of the GBA2000 burnt area processing.
- For date 26th February 2000 (20000226), the B3, MIR and VZA bands are not named in the conventional form. The user needs to extract and rename these files manually.
- For date 11th March 2000 (20000311), there are reported problems with spectral values in all bands. The data from this date is not used in the processing.
- For date 6th July 2000 (20000706), the dataset seems to be valueless for large regions of the globe. The user can still use this date but not for all regions.
- For date 27th July 2000 (20000727), the dataset seems to be valueless for large regions of the globe. The user can still use this date but not for all regions.
- For date 31st July 2000 (20000731), the dataset is completely absent from the data archive.
- For date 6th October 2000 (20001006), the MIR band does not decompress due to compression errors. Other decompression tools have been tried with no success. The problem was reported to the systems administrator. Without this band, this date cannot be processed by any GBA2000 algorithm and hence is not used.

3 Generic pre-processing module

This main objective of this module was to prepare the extracted data for the implementation of the burnt area algorithms. The objective of this data preparation stage was to remove, as much as possible, the likelihood that pixels would be falsely detected as being burnt areas. The main cause of false detection in the burnt area algorithms is the presence of cloud shadow pixels in the image. The data was also contaminated by tracks of saturated pixels in the MIR band, as a result of overloading the instrument's detection capacity. Also, problems were observed with the values of pixels located at the very edges of the swath. These edges could be removed, as the data covering this area would be collected away from the edge of the swath the following day. Obviously, there is the problem of cloud and cloud shadow contamination to solve. By accurately detecting those pixels affected by cloud, the removal of pixels affected by cloud shadow is made simpler. A further consideration is made to removing those pixels that are considered to be non-vegetated land surfaces, i.e. urban areas and water bodies. To enable this to be done in a consistent way, reference was made to the UMD land cover product previously described.

It became evident that many of the GBA2000 partners were attempting to solve the problems outlined above in different ways while effectively achieving the same results, or not having the time to work on the pre-processing requirements of the burnt area algorithm as well as the burnt area algorithm itself and producing average results because of this. After discussion with the GBA2000 partners to establish their broad pre-processing requirements, programs were developed that were flexible enough to be modified to each of the partner's requirements (i.e. different threshold values used) and specific enough so that each of the partners were satisfied with the pre-processed products. The result was that several of the GBA2000 partners adopted the general pre-processing module in their burnt area algorithm approach, whilst another used a modified version and a few others chose to develop their own pre-processing algorithms. In this chapter, the general pre-processing requirements are described in detail. For those burnt area algorithms that used a different pre-processing method, a description is given in the section in this report devoted to their algorithm. In any case, the reader is referred to the descriptions of the individual algorithms for thresholds specifications used when pre-processing the data. Further information and a description of all the code used in the pre-processing algorithms are available in Annex C.

A c-shell script has been developed to control the pre-processing of the extracted SPOT Vegetation S1 datasets. The script is called gba_preprocessor.csh. Typing this filename at the command line will yield information on how to use this program and the input variables required as shown here:

```
#########################################################
GBA2000 C shell: gba_preprocessor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_preprocessor.csh
<acronym>
<viewing zenith angle mask flag (1 = yes, 0 = no)>
<zenith angle threshold (degrees, e.g. 50), or '-'>
<mir saturation mask flag (1 = yes, 0 = no)>
<mir DN threshold (e.g. 1000), or '-'>
<cloud mask flag (1 = yes, 0 = no)>
<cloud mask operation (1,2,3,4 or '-')>
<cloud mask operator (0, 1 or '-')>
<1st input band (b0) cloud threshold (e.g. 200 or '-')>
<2nd input band (mir) cloud threshold (e.g. 180 or '-')>
<erode cloud mask by 1 pixel in 3x3 kernal (1 = yes, 0 = no)>
<shadow mask flag (1 = yes, 0 = no)
<height of cloud in metres (e.g. 10000), or '-'>
<memory (MBytes) available for shadow masking (e.g. 100), or '-'>
<non-burnable area mask flag (1 = yes, 0 = no)>
<apply mask to data level (select from: none, vza, mir, cloud, cl_comp, shadow, burn, basic or all)>
<delete mask (1,0) files (1 = yes, 0 = no)>
Notes:
If certain masking procedures are not implemented indicate variables with a '-'
View zenith angle mask can be applied to all products
MIR saturation mask threshold, mask = 1 if DN < threshold; otherwise 0
If MIR mask is present this will applied to the view zenith angle mask before shadow masking
For the cloud mask, type 'cloud_mask' at the command line for information
The shadow mask requires either a view zenith and cloud mask as input
The non-burnable area mask should be available in current directory
An information (text) should be present containing the geometry of the files
The 'cl_comp' apply level flag refers to cloud, vza, mir and non-burn masked products
The 'basic' apply level flag refers to vza, mir and non-burn masked products
Intermediate mask products will be deleted unless you specify 1 as the final variable
Files 'input_dates.txt' and '(acronym)_file_info.txt' must be available
```

The program is run in the working directory (e.g. ./AU). By entering values at the command line this c-shell script calls upon several different c programs. With this program, it is possible to apply all of the pre-processing algorithms available or just one of them; all that is required is to flag the relevant section of the input command to one and set the level to apply the pre-processing at the correct level. Input values are required at the command line to provide input to the c programs used. For example, if the user wanted to derive a mask of pixels contaminated by cloud, then the cloud mask flag would be set to one (1), threshold values would be specified and the cloud mask operation and operator would be set and the

apply level of cloud would be specified. A description of this c-shell script is available in Annex C.

As an example, the most common input command used in the GBA2000 project was to fully pre-process the original data. The output products would be screened for, extreme viewing zeniths (viewing angle zenith mask flag = 1, zenith angle threshold in degrees = 60), saturation in the MIR channel (MIR saturation mask flag = 1, MIR threshold = 1000), presence of cloud pixels (cloud mask flag = 1, cloud mask operation = 4, cloud mask operator = 1, B0 cloud threshold in DN = 180, MIR cloud threshold in DN = 180), erosion of the cloud masks by one pixel in all directions (erode cloud mask flag = 1), presence of cloud shadow pixels (shadow mask flag = 1, height of cloud in metres = 10000, memory allocation in Mbytes = 75) and the presence of non-burnable areas as defined by the UMD land cover product (non-burnable area mask flag = 1). The final step, after specifying the level of masking in the output product (apply mask level = all) and applying the masks to the original data, is to remove all of the intermediate composite images (delete masks = 1). An example of the input command that would fully pre-process a date from the Australian sub-window (AU) is:

gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 1 10000 75 1 all 1

The program would pre-process all of those dates specified in the file input_dates.txt. The name of the output files have the same naming structure and convention as the input files, except that they have an extension to their name indicating the level of pre-processing that has been applied (e.g. all, cloud, vza etc.). The outputs of the individual programs produce binary masks that are applied to the image data at the level requested by the user. It is these binary masks that are deleted after they have been applied to the original data and have no further use. For example the directory structure of one day for the Australian window would resemble the following after all of the pre-processing procedures have been applied.

./20000615/AU_20000615_b0
./20000615/AU_20000615_b0.hdr
./20000615/AU_20000615_b0_all
./20000615/AU_20000615_b0_all.hdr
./20000615/AU_20000615_b2
…
./20000615/AU_20000615_vza
./20000615/AU_20000615_vza.hdr
./20000615/AU_20000615_vza_all
./20000615/AU_20000615_vza_all.hdr

Each of the pre-processing procedures is now examined in detail. By doing this, it will also give the reader further instruction on the meaning of the various input parameters that are required in the controlling c-shell script. After these have been described, the different levels of pre-processing that can be applied to the data will be summarised.

Upon executing the program, checks are made for the presence of files needed for the programs to operate without problems. These are the files input_dates.txt and the text file containing the geometrical and projection information (e.g. AU_file_info.txt). If the data has been extracted with a buffer (of twenty pixels) surrounding the original data (see Section 2.5.2) then this needs to be taken into account as the correct geometrical information is contained within the file AU_buffer_file_info.txt for example. This does not cause any problems as the program tests for the presence of buffered products and renames the text files accordingly so that no confusion is made. The information file of the non-buffered window is renamed so that all of the processing is applied to the buffered window. When these tasks are complete the pre-processing of the data commences.

3.1 Masking of pixels acquired at extreme viewing zenith angles

The SPOT Vegetation instrument was designed so that the resampling of the data acquired for each pixel was uniform and was of an equal area on the ground. However, it was observed that the pixels at the edges of the swath were distorted and sometimes containing erroneous values. The viewing zenith angles (relative to the pixel on the ground) range reach values of 61 degrees either side of the central track of the satellite. Figure 4 shows an extract from the edge of a swath. The data is displayed as an RGB image (Red = band MIR, Green = band B3, Blue = band B2). It shows that there are some contaminated pixels at the edge of the swath, most likely caused by resampling of the data. A program was also needed to remove those pixels that were acquired at viewing zeniths greater than 50 degrees (for the IFI algorithm, see Chapter 5). Because of the frequent coverage of the SPOT Vegetation instrument, the algorithm developers decided that better quality data (those data acquired at smaller viewing zenith angles) was more useful than complete coverage of a region every day.

Figure 4: Example of the resampling errors observed at the edge of the satellite swath. A program was developed to remove all pixels acquired at a viewing zenith angle greater than a user specified value. The area shown is a small part of Australia.

The program developed satisfied both of these conditions by not fixing the threshold value of the viewing zenith angle, but rather demanding the value as an input to the program at the command line. The c program view_zenith_mask.c is used and requires the following input parameters:

view_zenith_mask
<input file (viewing zenith angles)>
<output file (1,0 mask)>
<pixels> <lines>
<threshold>

where, the output file is a binary (0/1) mask that is a result of writing a value of zero in the output file for all of those pixels with a viewing zenith angle value greater than the threshold value. All other values are assigned a value of one in the output file. The threshold value can be interpreted as a floating point value, although the resolution of the actual viewing zenith angles is only accurate to one half of a degree.

In the controlling c-shell script gba_preprocessor.csh input variables two and three (the first is the two letter acronym of the region the user is processing) indicate that the user wishes to generate the viewing zenith angle mask (viewing zenith angle mask flag = 1) and the value of the threshold to apply (zenith angle threshold = 60) as given in bold type here:

gba_preprocessor.csh AU **1 60** 1 1000 1 4 1 180 180 1 1 10000 75 1 all 1

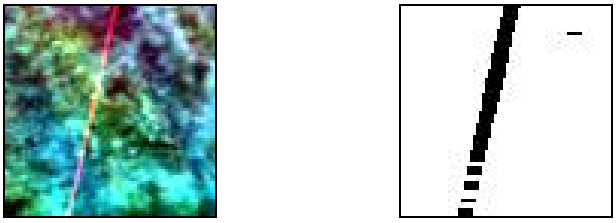
Inside the controlling c-shell script, an example of the command used to produce the viewing zenith angle mask is:

view_zenith_mask
$year$month$day/$acronym_$year$month$day_vza
$year$month$day/$acronym_$year$month$day_vza.mask
$pixels $lines
$vza_mask_threshold

A full description of this program is given in Annex C.

3.2 Masking of pixels affected by saturation in the MIR channel

Saturation in the MIR (middle-infrared, also known as the SWIR) channel is a common problem in the SPOT Vegetation S1 dataset. Some information on the location of the contaminated pixels is contained within the status file, situated on the data archive. However, after discussion with other users of this data, it was found that this information was not reliable enough to use as a guide to removing fully the offending areas. So another method was sought. As the name given to the problem suggests, the pixels have very high values (DN) in the MIR channel of the data. A simple method to remove them was to apply a threshold on the MIR channel, set high enough so that only the saturated pixels were removed. However, the problem was compounded in that because of the resampling method used for the S1 product, surrounding pixels would also be contaminated. The nature of this problem, when observed in an image, was that lines of saturated pixels would be present in the direction of the satellite's track. The resampling problem influenced up to two pixels either side of the saturated pixel. This problem is shown in Figure 5 for a region over northern Africa (the red line going north to south in the centre of the image). In the left image the saturated pixels are observed. In the right image, the offending pixels and those immediately adjacent to the saturated pixels have been removed.



(a) (b)

Figure 5: Example of contaminated pixels caused by saturation in the MIR channel of the SPOT Vegetation S1 sensor (a). The image on the right (b) is the result of the masking procedure being applied to the original data. The area shown is a small region in Africa.

The c program mir_sat_mask.c was developed to effectively mask out the contaminated pixels. The program requires a threshold value for the MIR channel set by the user at the command line. All of those pixels with a value in the MIR band greater than the threshold are set to a value of zero in the output image with all pixel values less than the threshold set to a value of one. Then, in a second step, those pixels situated two pixels to the left and the right are also masked. Because of the nature of the saturation lines observed in the image, it was

not necessary to mask pixels in any other direction. The program requires the following input parameters:

mir_sat_mask
<input file (MIR band)>
<output file (1,0 mask)>
<pixels> <lines>
<threshold>

where, all of those pixels with a MIR value greater than the threshold value are assigned a value of zero in the output image, other a value of one is assigned to the value of the pixel in the output file.

In the controlling c-shell script gba_preprocessor.csh, input variables four and five indicate that the user wishes to generate the MIR saturation mask (MIR mask flag = 1) and the value of the threshold to apply (MIR threshold = 1000) as given in bold type here:

gba_preprocessor.csh AU 1 60 **1 1000** 1 4 1 180 180 1 1 10000 75 1 all 1

Inside the controlling c-shell script an example of the command used to produce the MIR saturation mask is:

mir_sat_mask
$year$month$day/$acronym_$year$month$day_mir
$year$month$day/$acronym_$year$month$day_mir.mask
$pixels $lines
$mir_mask_threshold

A full description of this program is given in Annex C. The one noticeable problem with this program is that even if the threshold value is set very high, some very bright surfaces (deserts, salt pans etc.) are mistaken as saturated pixels. However, this was not a big concern as these regions are unlikely to be burnt due to a lack of vegetation.

3.3 Masking of cloudy pixels

The masking of pixels affected by clouds does not cause a direct problem with the mapping of burnt areas. Rather more, it is the cloud shadows that are projected from these clouds that cause the problems, as they have similar spectral properties to burnt areas. A number of techniques were considered for the detection of clouds and cloud shadows. These included studying the temporal signature of a pixel over a week, which will change because of the cloudy and cloud free conditions and looking at thresholds in the spectral bands. The requirements of some of the algorithm developers within the GBA2000 project, required cloud and cloud shadow free images on a daily basis. The approach taken was one outlined

by Kempeneers *et al*. (2000) at the SPOT Vegetation 2000 meeting in Italy. They used thresholds on channels B0 and MIR of the SPOT Vegetation sensor. However, the thresholds they derived were specifically for P product (raw data) and the same values did not work for the S1 products. The principle was utilised though and the thresholds used were set as variables at the command line by the user. The c program cloud_mask.c (Annex C) was written to detect the presence of clouds in daily S1 data. The input parameters are:

cloud_mask
<pixels> <lines>
<outfile (0,1 mask file)>
<input_band1 (e.g. b0)> <band1_threshold (e.g. 600)>
<input_band2 (e.g. mir)> <band2_threshold (e.g. 180)>
<operation (1,2,3 or 4)>
<logical_operator (AND = 0/OR = 1)>

where, the pixel in the output binary (0/1) file is assigned a value of one (not cloudy) subject to the value of the two band thresholds and the operation and operator flags. The operation to be undertaken on the two input files are indicated by one of four options, the output value being one if, operation flag = 1 signifies input pixel 1 AND/OR input pixel 2 are greater than both threshold values stated, operation flag = 2 signifies input pixel 1 is greater than the first threshold value AND/OR input pixel 2 is less than the second threshold value, operation flag = 3 signifies input pixel 1 is less than the first threshold AND/OR input pixel 2 is greater than the second threshold, operation flag = 4 signifies input pixel 1 AND/OR input pixel 2 are less than both threshold values. The logical operator flag indicates whether the conditions must both be satisfied (AND flag = 0) or either condition satisfied (OR flag = 1).

In the controlling c-shell script gba_preprocessor.csh input variables six to eleven indicate that the user wishes to generate the cloud mask (cloud mask flag = 1). The cloud mask will be generated, in this example, under the condition that a pixel is not cloudy if the value in the B0 band is less than 180 DN or if the value in the MIR band is less than 180 DN (cloud mask operation = 4 and cloud mask operator = 1). If these conditions for a pixel not contaminated by cloud are not satisfied then a value of zero is written to the output file. The eleventh parameter indicates that the cloud mask is to be dilated by one pixel in all directions using the generic c program erode_mask.c (Annex A). The input parameters to the controlling script that activate the cloud mask are:

gba_preprocessor.csh AU 1 60 1 1000 **1 4 1 180 180 1** 1 10000 75 1 all 1

Inside the controlling c-shell script, an example of the command used to produce the cloud mask is:

cloud_mask
$pixels $lines
$year$month$day/$acronym_$year$month$day_cloud.mask
$year$month$day/$acronym_$year$month$day_b0 $b0_threshold
$year$month$day/$acronym_$year$month$day_mir $mir_threshold
$cloud_mask_operation $cloud_mask_operator

After the cloud mask was produced, the cloud mask was dilated by one pixel in all directions to ensure that the edges of the clouds were removed from the data, as these pixels containing a mixture of cloud and land surface could still project shadows on to the land surface. The program used to dilate the cloud mask is the generic c program erode_mask.c (Annex A). The viewing zenith mask previously generated is used as the reference image in the program. The reference image indicates all of those pixels that contain values in the original data (i.e. not water and in between the swaths) and therefore are able to be masked out if adjacent to a cloud pixel. As an example, the inputs to the cloud dilation program are:

erode_mask
$year$month$day/$acronym_$year$month$day_cloud.mask
$year$month$day/$acronym_$year$month$day_vza.mask
$year$month$day/$acronym_$year$month$day_cloud.mask_temp
$pixels $lines

The following images show the sequence of cloud masking procedures applied to a small region in Australia. The data is displayed as an RGB image (Red = band MIR, Green = band B3, Blue = band B2). In Figure 6, the left image shows the original S1 product, with some obvious cloud contamination. The central image is the dilated cloud mask that has been calculated using the values given in the controlling c-shell script (B0 threshold = 180 DN, MIR threshold = 180 DN) and the right image is the product of the image data and the cloud mask to illustrate the effective removal of the cloudy pixels. The impact of cloud shadow on the spectral signature of the ground surface can be seen in these images. In the image on the right, there are burnt areas (displayed in dark purple to black colours as well as cloud shadow (displayed as dark areas) and making distinctions between these two phenomena are difficult to achieve on a daily basis. However, once the location of the cloud pixel has been established it is possible to project the likely position of the cloud shadow.

(a) (b) (c)

Figure 6: Example of the cloud masking procedure available in the GBA2000 pre-processing module. The original data is shown in the left image (a). The image in the centre is the dilated cloud mask (b) and the image on the right (c) is the product of the image data and the cloud mask. The area shown is a small region in Australia.

3.4 Masking of cloud shadow pixels

The removal of pixels that are contaminated by cloud shadows is a very important step in the pre-processing of data to derive the burnt area maps. If a land surface is mistakenly identified as being burnt, when actually under the shadow of a cloud, this has a very serious implication in that the pixel affected may not be analysed again during the implementation of the algorithms. This is certainly true for retrieving statistics on the final burnt area maps. Therefore, it is absolutely essential that all pixels likely to be cloud shadow be removed.

Previous research has shown that there are a number of ways to remove cloud shadows. These methods include monitoring a temporal sequence of signatures to determine when the pixel is in shade, applying thresholds to the spectral data, making a supervised or unsupervised classification of the image and calculating the projection of the cloud shadow (knowing the position of the cloud, the illumination and satellite viewing conditions and the height of the cloud). Unfortunately, there was no way of deriving the cloud height from thermal measurements because such an image band was not available on the SPOT Vegetation sensor. Another factor that was considered when deciding upon a cloud shadow masking algorithm was the computational demands the program would make on memory, disk space and CPU time. Given the large dimensions, in space and time, of the SPOT Vegetation dataset, something needed to be developed that would be fairly quick and efficient to implement.

The method chosen was a modified version of an algorithm developed for the detection of cloud shadows developed by Kempeneers *et al*. (2000). The modifications made to the original algorithm significantly reduced the processing time and ensured that all cloud shadow pixels would be detected, even if that meant the loss of some information by exaggerating the amount of cloud shadow. The algorithm is available as a c program called shadow_mask.c. The input parameters to the program are:

shadow_mask
<cloud_mask (byte)>
<viewing zenith mask (byte) (or good data mask)>
<output_mask (byte) (e.g. shadow mask)>
<viewing_zenith_angles (vza)> <sun_zenith_angles (sza)>
<viewing_azimuth_angles (vaa)> <sun_azimuth_angles (saa)>
<pixels> <lines>
<image_start_lon> <image_start_lat (of input image coordinate (0,0) in degrees)>
<cloud height (in metres, e.g. 10000)>
<memory available for processing (MBytes)>

A description of the input parameters is made easier by describing the individual stages of the program. The pixels that were detected as being clouds (after dilation) are read in by the program (as a binary 0/1 mask). In addition, a reference image (also a binary 0/1 mask) containing all of those pixels that can possibly be cloud shadow (or land surface) is also read in. In this case the mask used was that produced by the viewing zenith angle masking program. To calculate the position of a cloud, information on the illumination geometry and the viewing geometry of the satellite is used. This information is provided to the program by reading in all of the four angle bands.

To calculate the projection of a cloud shadow onto the Earth's surface given the position of a cloud requires the latitude and longitude of the cloudy pixel to be known. This can be calculated by knowing the latitude and longitude of the upper left pixel of the image being processed. This information is derived from the sub-window text file located in the working directory. The projection of the cloud shadow onto the Earth's surface is also dependent on the height of the cloud. In the GBA2000 cloud shadow program it is this assumption, i.e. the height of the cloud, which is different from the original algorithm by Kempeneers *et al*. (2000). In the original algorithm spectral signature tests were made along the line of sight from the cloud pixel to where the cloud shadow should be located. When the spectral signature shifted by a certain amount, this indicated that the pixel was not under a cloud shadow. The pixel offset between the cloud pixel and the pixel with the non-shadow spectral

signatures was calculated in terms of a cloud height. This cloud height estimate was then applied in a local area to remove all cloud shadow. GBA2000 required something a little more simple and practical. In theory clouds rarely reach higher than 10 km. In reality, most clouds exist at a height much lower than this. However, as a compromise between the need to be sure of removing all cloud shadow pixels and the rare occurrence of clouds higher than 10 km, a cloud height of this value was assumed for all regions across the globe. During testing, the algorithm was observed to overestimate the areas of cloud shadow in many circumstances. However, also observed were times when the cloud height assumption was perfect. It was too time consuming to calculate average cloud heights for different regions and would eventually be as worthless task as most cloud types occur in all regions of the globe. However, it was confirmed that for pre-processing the global S1 dataset, this method was very efficient at removing cloud shadow pixels over large regions. To enable the user to test different cloud heights, or if the user is working on a very small region with specific cloud height information, the cloud height value was made an input parameter to the c program shadow_mask.c.

The final parameter in the cloud shadow program is the amount of memory that the user can afford to devote to the cloud shadow masking procedure. Because cloud shadows can be projected in any direction, in terms of processing, this means that the entire array needs to be read into the memory of the workstation. For a large area (e.g. continental Africa), the reading of seven large files the size of Africa to memory may cause the machine to crash. To solve this problem, and therefore be able to process large areas, the user specifies the amount of memory available on their machine. If the amount of memory is less than the total amount of memory needed by the program then the sub-window is separated into components, each with an overlap region, that are processed separately and then joined back together again. This separation and then adding back together of regions within the sub-window is all carried out within the program.

The program firstly determines those pixels that are assigned as being cloudy (a value of zero in the cloud mask) but have been assigned as being available data (a value of one in the viewing zenith angle mask). From the angle bands the actual angles are calculated and then converted to radians (instead of a DN represented in byte format). The geometric model for the determination of the cloud and cloud shadow vector is shown in Figure 7. The cloud pixel (p) is located at the centre, though the real cloud is at height h from the tangential plane (the

intersection of the sunbeam and the line of sight of the satellite/cloud pixel). The shadow pixel can be found at the intersection of the sunbeam and the tangential plane at the centre. The solar zenith and solar azimuth angles are assumed equal in both cloud and shadow pixels. Let φ be the angle between the meridian north (taken as the X-axis) and the vector cloud-shadow pixel. From Figure 7, it is shown that φ equals the sum of γ (positive or negative according to relative azimuth angle) and the viewing azimuth angle (Kempeneers *et al.* 2000).

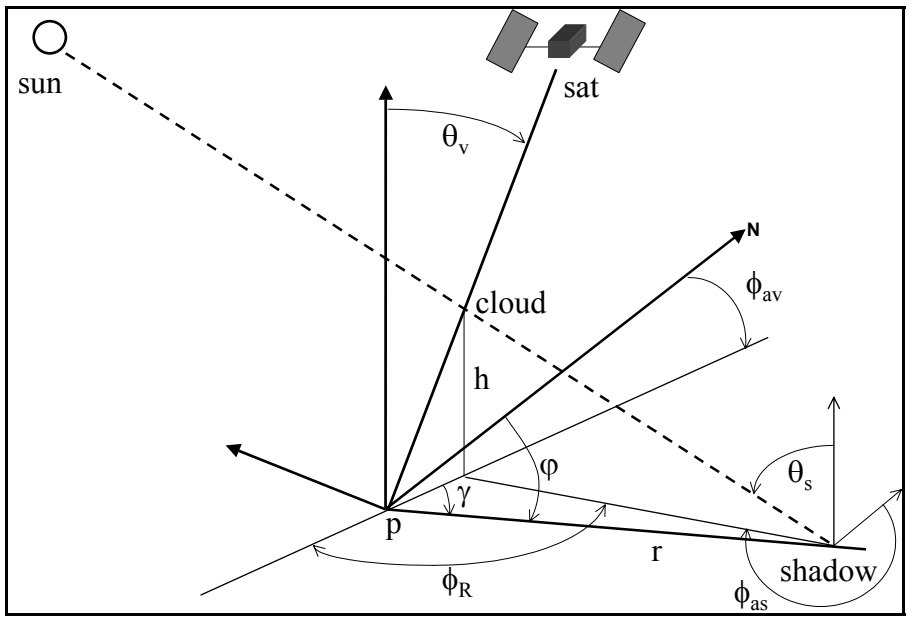


Figure 7: The geometrical model used to calculate the location of the cloud shadow pixel given the location of the cloud pixel and the illumination and viewing geometries.

The position of the shadow pixel is fixed when r and φ are known. Using geometry, they are calculated as:

$$r = h\sqrt{\tan^2\theta_s + \tan^2\theta_v - 2\tan\theta_v \tan\theta_s \cos\phi_R}$$

$$\varphi = \underbrace{\pm \arccos\left(\frac{\tan\theta_v - \tan\theta_s \cos\phi_R}{\sqrt{\tan^2\theta_s + \tan^2\theta_v - 2\tan\theta_v \tan\theta_s \cos\phi_R}}\right)}_{\gamma} + \phi_{av}$$

where, a positive sign is taken for relative azimuths ($\phi_R = \phi_{av} - \phi_{as}$) between $0 \le \phi_R \le \pi$, and a negative sign is taken for relative azimuths between $\pi < \phi_R < 2\pi$. ϕ_{av}, ϕ_{as} are the viewing and solar azimuth angles and θ_v, θ_s are the viewing and solar zenith angles respectively. Notice that φ is independent of the cloud height.

The straight line defined by φ in the tangential plane corresponds to a defined curve on earth. The possible positions for the real shadow pixel will lie on this curve. The corresponding pixels in the image will be referred to as potential shadow pixels. With the angle φ fixed, the only degree of freedom in the position of the shadow is the distance between cloud and shadow pixel (a function of the cloud height). A simplified approach is to define the position of the shadow in the tangential plane on earth:

$$longitude_2 = longitude_1 + \frac{r \sin\varphi}{\cos\left(latitude_1 + \frac{r\cos\varphi}{2}\right)}$$

$$latitude_2 = latitude_1 + r\cos\varphi$$

where, r is the distance between cloud and cloud shadow in Nautical Miles (1Mn=1,852 km). As a result, the second term containing r is in minutes.

The next step is to calculate the distance r, given that the cloud height is specified at 10,000 m (or a value stated in the command line). From this situation it is possible to calculate the latitude and longitude of both the cloud pixel and the position where the shadow pixel would be situated under these conditions. From the position of the latitude and longitude of the shadow, this is converted to the nearest equivalent pixel and the offset (in pixels) between cloud and shadow pixel (from a projected height of 10 km) calculated. In theory, all those pixels between the cloud pixel and the shadow pixel can be contaminated with cloud shadow. Therefore, all of those pixels in a line between the cloud and shadow pixel are masked out. The program writes an output file assigning a zero value to pixels calculated to be cloud shadow and a value of one assigned for data that is clear from cloud shadow. A full description of the program code is given in Annex C.

Figure 8 shows the result of generating a cloud shadow mask for the same area shown in Figure 6. On the left is the cloud masked data, in the centre is the shadow mask and on the right is the result of applying the shadow mask to the original data. Although some data not contaminated with cloud shadow are lost, the algorithm ensures that no shadows remain in the image and therefore the likelihood of cloud shadow pixels being detected as burnt pixels is significantly reduced.

(a) (b) (c)

Figure 8: Example of the cloud shadow masking procedure available in the GBA2000 pre-processing module. The original data that has been masked for cloud is shown in the left image (a). The image in the centre is the cloud shadow mask (b) and the image on the right (c) is the product of the image data and the cloud shadow mask. The area shown is a small region in Australia.

In the controlling c-shell script gba_preprocessor.csh, input variables twelve, thirteen and fourteen indicate that the user wishes to generate the cloud shadow mask (shadow mask flag = 1), calculated from a cloud height of 10 km (height of cloud in metres = 10000) using a memory allocation of 75 Mbytes. The input parameters to the controlling script that activate the cloud shadow mask are shown in bold type here:

gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 **1 10000 75** 1 all 1

Inside the controlling c-shell script, an example of the command used to produce the cloud mask is:

shadow_mask
$year$month$day/$acronym_$year$month$day_cloud.mask
$year$month$day/$acronym_$year$month$day_vza.mask
$year$month$day/$acronym_$year$month$day_shadow.mask
$year$month$day/$acronym_$year$month$day_vza
$year$month$day/$acronym_$year$month$day_sza
$year$month$day/$acronym_$year$month$day_vaa
$year$month$day/$acronym_$year$month$day_saa
$pixels $lines
$start_lon $start_lat
$cloud_height
$memory

The program code for the cloud shadow masking procedure is presented in Annex C.

3.5 Masking for water and non-vegetated land surfaces

The removal of pixels that are representative of non-vegetated surfaces was carried out for a number of reasons. These were to reduce the amount of data that was to be processed, to remove water bodies that were not screened in the original S1 products and to remove areas

that could be mistaken as burnt areas in some of the algorithms (shore lines and urban regions). The UMD land cover product was used to derive the information required to produce the mask. The mask for the region of interest was extracted from the global dataset in the data extraction module. For cover types water, urban and built-up, and bare ground, a value of zero was assigned to the pixels in the mask and all other pixels were assigned a value of one. In the controlling c-shell script gba_preprocessor.csh, input variable fifteen indicates that the user wishes to apply the land cover mask (non-burnable area mask flag = 1). The input parameters to the controlling script that activate the land cover mask are:

gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 1 10000 75 **1** all 1

3.6 Pre-processing module output product description

The c-shell script, gba_preprocessor.csh controls the production of masks and masked output products. The final two parameters of this script control the level of output product the user requires and whether the user wants to keep on disk copies of the binary mask products (instead of just the masked spectral and angle bands). The c-shell script makes use of the generic c program, apply_mask.c (Annex A) to derive the product of the spectral (or angular) data and the binary (0/1) masks. The user can select the following levels of mask to be generated and applied to the original data. Note that when a particular process is not required the input variables are replaced by a dash (-):

gba_preprocessor.csh AU 0 - 0 - 0 - - - - 0 0 - - 0 **none** 0

where, the input parameter none indicates that the user wishes to derive output products that have not been masked for any of the properties mentioned previously. This just results in a name change of the input files.

gba_preprocessor.csh AU 1 60 0 - 0 - - - - 0 0 - - 0 **vza** 1

where, the input parameter vza indicates that the user wishes to derive output products that have been masked for extreme viewing zeniths only.

gba_preprocessor.csh AU 0 - 1 1000 0 - - - - 0 0 - - 0 **mir** 1

where, the input parameter mir indicates that the user wishes to derive output products that have been masked for saturation in the MIR channel only.

gba_preprocessor.csh AU 0 - 0 - 1 4 1 180 180 0 0 - - 0 **cloud** 1

where, the input parameter cloud indicates that the user wishes to derive output products that have been masked for cloud only. Note that dilation of the cloud mask can only be undertaken if the viewing zenith angle mask has also been created.

gba_preprocessor.csh AU 0 - 0 - 0 - - - - 0 0 - - 1 **burn** 1

where, the input parameter burn indicates that the user wishes to derive output products that have been masked for non-vegetated land cover only.

gba_preprocessor.csh AU 1 60 1 1000 0 - - - - 0 0 - - 1 **basic** 1

where, the input parameter basic indicates that the user wishes to derive output products that have been masked for extreme viewing zenith angles, saturation in the MIR channel and non-vegetated land covers.

gba_preprocessor.csh AU 1 60 0 - 1 4 1 180 180 1 1 10000 75 0 **shadow** 1

where, the input parameter shadow indicates that the user wishes to derive output products that have been masked for cloud shadow. In this case, the output products are also masked for extreme viewing angles (the value set by the user) and clouds, as the cloud shadow program requires these products.

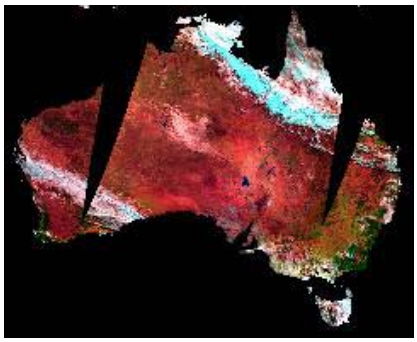
gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 0 - - 1 **cl_comp** 1

where, the input parameter cl_comp indicates that the user wishes to derive output products that have been masked for extreme viewing zenith angles, saturation in the MIR channel, cloud (and in this case dilation of the cloud mask) and non-vegetated land covers.

gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 1 10000 75 1 **all** 1

where, the input parameter all indicates that the user wishes to derive output products that have been masked for extreme viewing zeniths, saturation in the MIR channel, cloudy pixels (followed by dilation by one pixel of the cloud mask), cloud shadow and non-vegetated land cover and water. In all of the above cases, the intermediate binary mask files are deleted after the program has finished.

The output files have the same naming convention as the original input files, except with an extension indicating to the user the level of pre-processing applied to the data. For example, those data pre-processed with all of the available programs will contain the word 'all' in the output file. An example of the image outputs from the pre-processing module is now shown for continental Australia (Figure 9). The first image (a) is a RGB image (Red = band MIR, Green = band B3, Blue = band B2) collected on the 15th June 2000. Cloud can be seen in the north and south west of the image. Masks shown are of extreme zenith viewing angles in (b), saturation in the MIR channel (note that some bright desert surfaces are also masked out) in (c), the cloud mask, that has been dilated by one pixel (note again that some pixels indicating bright, desert surfaces, have also been masked as being cloudy pixels) in (d), the cloud and cloud shadow mask in (e) and the non-vegetated land cover mask shown in (f). When all of these masks are applied to the original dataset the output products indicate those pixels that are of sufficient quality to be considered for the detection of burnt areas. For the example, shown in Figure 9, those pixels remaining after the pre-processing module are shown in (g).



(a) Original S1 RGB image

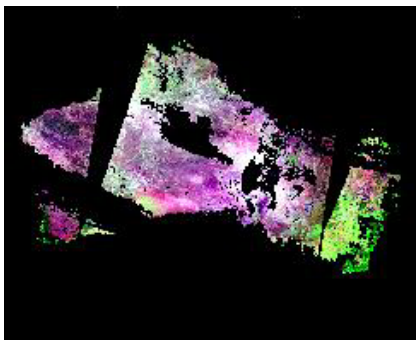
(b) Viewing zenith angle mask

(c) MIR saturation mask

(d) Cloud mask (with dilation)

(e) Cloud shadow masked

(f) Non-vegetated land cover mask



(g) Pre-processed S1 product

Figure 9: Example of the full application of the pre-processing module developed for the GBA2000 project. An original S1 product for 15th June 2000 is shown in (a). The masking procedures are applied to the data to produce the pre-processed output image (g).

4 Image data compositing module

The objective of deriving image composites is to produce an image product that is of a better quality than each of its individual components. The use of the word, better in the previous sentence is used to cover a number of situations those being, for example, cloud free products, cloud shadow free products or images without the spatial heterogeneity of daily images. The use of composites to derive information about biophysical properties of land surfaces has been widely published (Lissens *et al*., 2000; Duchemin *et al*., 2000; de Wasseige *et al*., 2000). It is beyond the scope of this report to provide specific details of the scientific background of published compositing methods. However, a number of the GBA2000 partners use composited SPOT Vegetation S1 products from which the burnt area products are derived. Therefore, the method of generating these composites within the GBA2000 project is described.

An image composite is derived from individual images according to specific criteria that allow the most suitable pixel to be chosen from the individual datasets to construct the composite. The requirements of the GBA2000 partners also demanded that the time period of that the individual images represented needed to be flexible (i.e. from ten days to one month). Extensive testing of the quality of composites of pre-processed (according to algorithms described in Chapter 3) daily data compared with original S1 products was made. GBA2000 partners specified two compositing criteria as a component of their burnt area algorithms. The first criteria is based on the minimum reflectance in the near-infrared channel (B3), abbreviated in this report as minNIR; and the second based on the maximum normalized difference vegetation index (NDVI), abbreviated in this report as maxNDVI. A third algorithm, based on the minimum near-infrared method, that allows the user to select which value in the time period is selected to construct the composite is also described.

4.1 MinNIR compositing method and programs

The compositing criteria based on the pixel with the lowest value in the SPOT Vegetation channel B3 (near-infrared), the minNIR, was used as it was believed that if a pixel was burnt then that pixel would be chosen to be represented in the composite due to the changes in the spectral signature of that pixel. Problems associated with this method include the possible inclusion of cloud shadow pixels in the final composite, as these pixel values are also lower

than background vegetation in the B3 channel. Also, pixels that are permanently cloudy or contaminated by some other phenomena throughout the whole of the compositing time period will be included in the compositing product. Because of these concerns, tests were made on the quality of the output composites using S1 products that had been pre-processed prior to compositing and deriving composites and associated pixel status maps according to a temporal calculation made on the series of near-infrared values in the compositing period. The algorithm was developed initially by Stroppiana and Gregoire (2001).

To produce the minNIR composites a c-shell script called gba_min_nir_composite.csh was written. If you type this program name at the command line then the following instructions are given:

```
#########################################################
GBA2000 C shell: gba_min_nir_composite.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_min_nir_composite.csh
<acronym>
<path to gba_jrc_minnir_composite_angles.pro>
<path to IDL executable binary>
<apply mask to data to which level (select from: none, vza, mir, cloud, cl_comp, shadow, burn or all)>
<time period for composite (monthly = 'm', ten day = 'd'>
<output level 2 products: day of burn, cloud free days and angles (0 = no, 1 = yes)>
```

where, the acronym is the two letter window code, the full paths to the IDL program that makes the compositing and the IDL binary are stated, the apply mask parameter indicates to the program the level of pre-processing that the user requires to be composited (the codes point the program to the correct input files that have been produced by the pre-processing algorithms, presented in Chapter 3), the time period for compositing indicates to the program those daily datasets the user would like to be composited (a d here indicates that decadel (ten day) composites are to be made, a m indicates both monthly or non-decadel composites are required) and the final parameter indicates that the user would like, what are called here, level two composite products (binary products of, the first day in the period of daily dates from that the composite pixel was selected, the number of cloud free days in the compositing period and composites of the angle bands), as opposed to level one products that are just the composite images of the spectral data. Further details of this program are given in Annex D.

The time period that is to be composited is controlled by the text file input_dates.txt. The data directories should be organised in the structural form outlined in Section 2.1. The c-shell

script is run in the directory where the individual date directories are listed. The ten-day periods normally listed in the literature are from the 1st to the 10th, the 11th to the 20th and the 21st until the end of the month. If a month's worth of data is available on the disk and the user requires three ten day composites to be produced then the file input_dates.txt should be edited to reflect the processing for the whole month and the flag d chosen at the command line input. If the user requires a monthly composite then the flag m is chosen at the command line input. If the user requires a composite for a particular time period (for example between the 6th and 15th day of a month) then the file, input_dates.txt is edited for this period and the flag m chosen at the command line. It is also possible to composite images that cross over into different months, but this requires some file name changes and extension of the file input_dates.txt to dates that do not in theory exist (i.e. 20000732, 200000733 etc.). In the example shown below, a level two, ten day composite (20000101 to 20000110) is produced from input data that has been fully pre-processed according to the algorithms described in Chapter 3. The command line input to composite the data is:

gba_min_nir_composite.csh AU ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin all d 1

The first step of the program is the creation of a directory, in which the composites will be written. The name of the directory corresponds to the compositing period. In the case of the selecting m as the compositing flag then the directory created is always named with the extension *01_31. In this example, the directory 20000101_10 is created. Because the input data have been selected as being pre-processing, then these files must be present within the individual date directories otherwise the program will fail to work.

The compositing program has been coded in IDL and is called automatically from commands within the c-shell script. The c-shell creates band sequential (BSQ) data stacks of the input data. In the example shown, the user has selected the production of level two products, therefore the data stacks will consist of both spectral data and angular data (as opposed to solely spectral data for the level one composites). The BSQ files are created using the standard SUN OS command, cat. After the BSQ files have been created for each day in the compositing period, two temporary text files are automatically created that are needed by the IDL program. The first file, single_files_input_list.txt contains a list of the names of all the BSQ files available in the compositing directory (in doing this, if a date is unavailable or missing such as on 20001006 then the program knows not to look for this data because it is not listed in this text file). The second file called decade_files_input_list.txt just contains the

name of the text file single_files_input_list.txt. This is required if multiple composites are to be constructed within the IDL program itself. Instead, the decision to derive multiple image composites (i.e. 20000101_10, 20000111_20 etc.) is made outside the IDL by editing the file input_dates.txt and the command line input accordingly.

Two IDL programs have been written. The first called gba_jrc_minnir_composite.pro produces composites of the spectral data only. The second produces the level two products and is called gba_jrc_minnir_composite_angles.pro. Only the second IDL program is described here, as the working of the first algorithm is one component of the second. The full code of the IDL program is given in Annex D.

The IDL program reads in and processes the data on a line-by-line basis. For each pixel the non–zero values of the pixels in the B3 channel (near-infrared) are sorted. The presence of zero values in the datasets indicate either no data in the original dataset or data masked out in the pre-processing phase. The number of non-zero pixels is termed in this program as the number of cloud free days. If the number of cloud free day's equals zero then the respective pixel in the output composite is also zero. If the number of cloud free days is greater than three (>) then a temporal condition is tested for in the data. This temporal condition provides another defence against the inclusion of cloud shadow pixels in the output composites. The mean value of the second, third and fourth lowest pixel values is calculated. Then the range value between the fourth and second lowest pixels values is calculated. The lowest B3 (minNIR) pixel value is written to the output composite only if its value is greater than the mean statistics calculated minus the range statistic calculated. This indicates whether the lowest value is either an extreme or a singular occurrence. A byte array is created that gives the user some extra information about the selection process of the composite pixel. In this case the value of the pixel in the status binary is three. This indicates that a good selection of cloud free days is available and the lowest value conforms to the temporal condition specified above. If the temporal condition is not satisfied and the lowest minNIR value was selected on the first day of cloud free data then this pixel is assigned a value of two in the status array. This indicates to the user that the composite value was selected on the first day of cloud free data. Otherwise a value in the status array of one is assigned. In cases where the temporal condition is not satisfied, the second lowest pixel is used to create the composite (indicated by values of two and one in the status image array). An example may clarify this concept. In the case a pixel burns the last day of the compositing period, assuming that at least four clear

observations are available, the temporal test would not be satisfied and a day in which the pixel is not burnt would be selected as the composite day. The consecutiveness test would not be satisfied because the minimum near-infrared value occurred on the last day and hence the pixel would be flagged with a value of one. The flag value does not allow retrieving the burnt pixels that were discarded due to the temporal test but may help in the validation stage for identifying problematic pixels. If the number of cloud free days is three or less (excluding zero) then the lowest value on the B3 channel is selected for the composite and a zero value written to the status image array.

Once the selection of the pixel with the lowest minNIR value is made, data is extracted from all of the images associated with this pixel and date. In addition, to the composites and the status image, an image with pixel values indicating the day within the compositing period from which the pixel was selected (expressed as a integer value between one and n, where n is the day of the final input date) is created. An image with pixel values corresponding to the number of cloud free days is also created. Using the example given above, the composite directory structure after completion of the compositing process will resemble this:

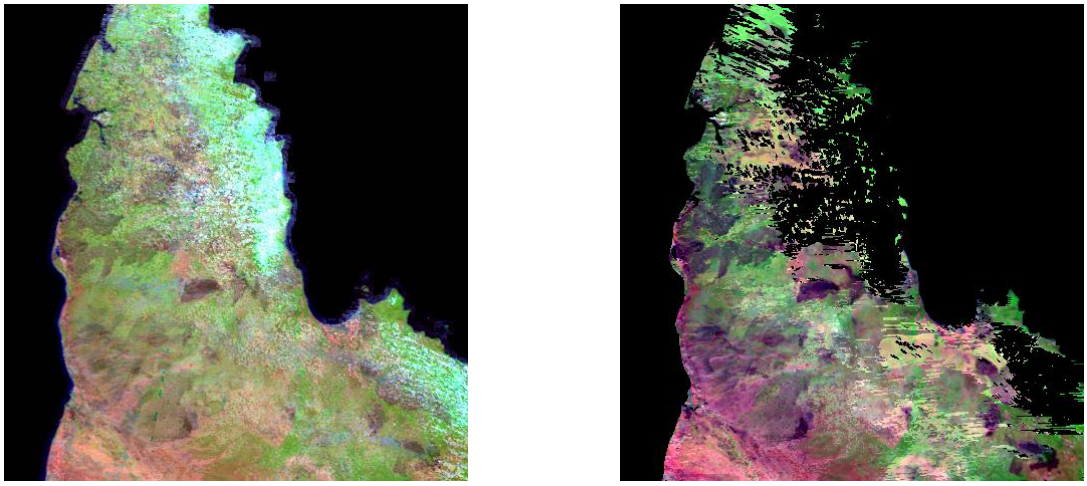
```
./20000101_10/AU_20000101_10_b0_all_minNIR
./20000101_10/AU_20000101_10_b0_all_minNIR.hdr
…
./20000101_10/AU_20000101_10_vza_all_minNIR
./20000101_10/AU_20000101_10_vza_all_minNIR.hdr
./20000101_10/AU_20000101_10_cfr_all_minNIR
./20000101_10/AU_20000101_10_cfr_all_minNIR.hdr
./20000101_10/AU_20000101_10_day_all_minNIR
./20000101_10/AU_20000101_10_day_all_minNIR.hdr
./20000101_10/AU_20000101_10_flag_all_minNIR
./20000101_10/AU_20000101_10_flag_all_minNIR.hdr
```

where, the acronyms cfr, day and flag, refer to the files indicating the number of cloud free days, the day in the compositing period that the pixel with the lowest minNIR was extracted, and the status image respectively. The file names indicate that pre-processing has firstly been undertaken.

4.1.1 Comparison of pre-processing S1 and original S1 composites

The following figures show the improvements that are gained by pre-processing the S1 data prior to compositing. Figure 10 shows an RGB image (Red = band MIR composite, Green = band B3 composite, Blue = band B2 composite) of the result of the minNIR compositing criteria for a ten-day period for a small region of north-east Australia. The image on the left has been composited without any pre-processing, the image on the right has been full pre-

processed for contaminated pixels. Although there is data missing in the pre-processed composite, where permanent cloud exists, there is a much lower likelihood that cloud shadow pixels are present in the composited images.



(a) (b)

Figure 10: Example of the improvements in the image composite as a result of pre-processing the data prior to compositing. Though the data quantity is reduced the likelihood that contaminated pixels are considered to be burnt areas is significantly reduced. The composited image without any pre-processing is shown in the left image (a), the fully pre-processed composite is shown on the right (b). The area shown is a region in north-east Australia.

In the next set of examples, small extracts of data have been made from original and pre-processed composites for decadal and monthly compositing periods. The images show that by compositing with pre-processed data, the 'salt and pepper' noise is removed from the composites, permanently cloudy areas (over the compositing period) are removed, cloud shadows are removed and burnt areas are more visually distinguishable (with similar histogram stretching). Figure 11 displays some of the results of compositing pre-processed data.

The specific use of the minNIR compositing criteria to produce image composites for the determination of burnt areas is discussed within the chapters of this report that describes the individual burnt area algorithms developed by the GBA2000 partners.

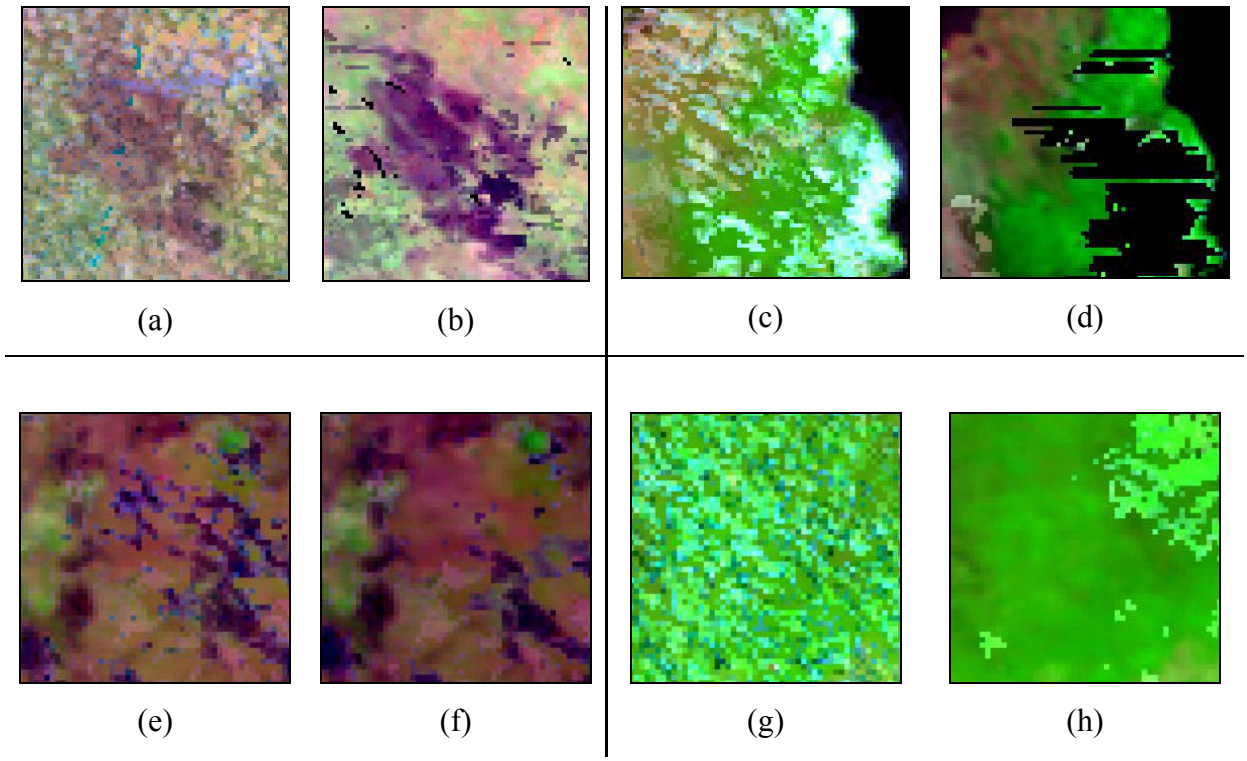


Figure 11: Example of the improvements in the image composite as a result of pre-processing the data prior to compositing. (a), (b), (c) and (d) are ten day composites of a small region in Australia, (e), (f), (g) and (h) are monthly composites of a small region in Mozambique, Africa. (a) and (b) show that pre-processing the data reduces noise levels over burnt areas in the composite. (c) and (d) show that improvements of the composite product over tropical forest areas are made by reducing the cloud contamination. (e) and (f) show that cloud shadow contamination is removed from the composite product after pre-processing. This is also the case over a region that suffers from cloud and cloud shadow contamination as shown in (g) and (h).

4.2 MaxNDVI compositing method and programs

The compositing of pixels based on the maximum value of the normalized difference vegetation index (maxNDVI) over the compositing period is well practiced. In fact, the method is used to produce the standard S10 products available from the SPOT Vegetation system. The procedure to derive the composite products is similar to that outlined in the previous section. A c-shell script called gba_max_ndvi_composite.csh has been written to produce the composites. The user can specify the level of pre-processed data to be composited (as described in the previous section). The user is able to generate composites of ten days, monthly or a user-specified time interval. If you type only the file name at the command line then the following instructions are given:

```
#######################################################
GBA2000 C shell: gba_max_ndvi_composite.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_max_ndvi_composite.csh
<acronym>
<path to gba_ccrs_maxndvi_composite.pro (IDL composite program)>
<path to IDL executable binary>
<apply mask to data to which level (select from: none, vza, mir, cloud, cl_comp, shadow, burn or all)>
 <time period for composite (monthly = 'm', ten day = 'd'>
```

where, the input parameters required by this program are identical to those discussed in Section 4.1. A full description of this program is given in Annex D.

The NDVI provides an indication of the presence of photo-synthetically active vegetation within the area covered by a pixel. Therefore, deriving the composite of the maximum NDVI maximises the likelihood that the pixel selected is not cloud or cloud shadow but indeed vegetation. If the pixel is burnt then the NDVI is reduced, but if it is low for the whole of the compositing period then the pixel value in the composite will also be low. An IDL program has been written that generates the composite products, for both spectral and angle bands based on this criterion. The IDL program gba_ccrs_maxndvi_composite.pro was written to make the composite processing under this criterion. The IDL program reads in temporarily created BSQ data stacks. For each pixel in each day of the compositing time period, the NDVI is calculated, from the ratio of the near-infrared (B3) value minus the red (B2) value over the near-infrared (B3) value plus the red (B2) value. For each pixel, the day in the compositing period, when the maximum value of the NDVI occurred was extracted for all of the spectral and image bands. In addition to the four spectral bands and the four angle bands, four more files are written to disk. These files are, first, an image of the number of cloud free days observed in the compositing period, second, an image with pixel values indicating the day within the compositing period on which the maximum NDVI value occurred, third, a image of the maximum NDVI values and, fourth, an image of the values of the short-wave vegetation index (SWVI) based on pixels selected under the maxNDVI criteria. The SWVI is derived from the ration of the near-infrared (B3) value minus the middle-infrared (MIR) value over the near-infrared (B3) value plus the middle-infrared (MIR) value. As well as providing useful information on the performance of the compositing algorithm the NDVI and SWVI products are also input data required by algorithms developed to determine burnt areas.

The output directory structure and composite products are named in a similar convention to the composites produced by the minNIR criteria, depending on the compositing time period and the level of pre-processing previously applied to the data. There was no detailed analysis of the advantages of deriving the composites with pre-processed S1 data as opposed to original S1 data undertaken for this compositing criterion. The reader is referred to Chapter 9 for examples of images composited under the maxNDVI criteria.

4.3 NIR value compositing method and programs

The compositing criteria based on the pixel with a selected value in the SPOT Vegetation channel B3 (NIR) was required by the joint Portuguese and Brazilian partner for the window BR in Figure 3 (see Chapter 11). To produce the NIR composites based on a selected value of the NIR, a c-shell script called gba_nir_value_composite.csh was written. If you type this program name at the command line then the following instructions are given:

```
#########################################################
GBA2000 C shell: gba_nir_value_composite.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_nir_value_composite.csh
<acronym>
<path to gba_nir_value_comp_angles.pro>
<path to IDL executable binary>
<apply mask to which level (select from: none, vza, mir, cloud, cl_comp, shadow, burn, basic or all)>
<time period for composite (monthly = 'm', ten day = 'd'>
<value of the NIR band (ordered low to high) to extract (3 = 3rd lowest)>
```

where, the acronym is the two letter window code, the full paths to the IDL program that makes the compositing and the IDL binary are stated, the apply mask parameter indicates to the program the level of pre-processing that the user requires to be composited (the codes point the program to the correct input files that have been produced by the pre-processing algorithms, presented in Chapter 3), the time period for compositing indicates to the program those daily datasets the user would like to be composited (a d here indicates that decadel (ten day) composites are to be made, a m indicates both monthly or non-decadel composites are required) and the final parameter indicates the ordered value of the NIR band that will be used to construct the composite image. As an example, a value of three here would indicate that the third lowest value (ignoring zero) from a months worth of daily data would be used to make the composite. The program automatically creates the level two composite products (comprising of images of the first day in the period of daily dates from that the composite

pixel was selected, the number of cloud free days in the compositing period and composites of the angle bands). Further details of this program are given in Annex D.

The time period that is to be composited is controlled by the text file input_dates.txt. The data directories should be organised in the structural form outlined in Section 2.1. The operation of this program is exactly the same as that described previously in Section 4.1. The command line input to composite the data is:

gba_nir_value_composite.csh BR ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin basic m 3

The first step of the program is the creation of a directory, in which the composites will be written. The name of the directory corresponds to the compositing period. In the case of the selecting m as the compositing flag then the directory created is always named with the extension *01_31. The compositing program has been coded in IDL and is called automatically from commands within the c-shell script. The IDL program is named gba_nir_value_comp_angles.pro and produces level two products. The full code of the IDL program is given in Annex D. The operation and output products of this compositing program are similar to those described in Section 4.1. The algorithm is described in detail by Cabral *et al*. (accepted to the IJRS).

5 IFI algorithm implementation module

Colleagues at the International Forest Institute (IFI) in Moscow, Russia (Ershov and Novik, 2001) provided an algorithm for burnt area detection in forest and non-forest regions of Russia. The algorithm consists of three main parts:

- Masking of contaminated pixels.
- Detection of pixels that are possible burnt areas and creating of an intermediate composite reference image.
- Correction of a potential burnt area mask.

The algorithm utilises daily S1 products from SPOT Vegetation. To take into account large-scale temporal and spatial variations in land cover and characteristics in Russia, the algorithm uses statistical information over blocks approximately 200 by 200 km in size. This minimises the influence of factors such as forest changes during spring and autumn periods and the histograms of non-forest classes. In addition, the IFI algorithm makes a distinction between grassland surfaces and non-grassland surfaces (e.g. forest, woodland and cropland) in assigning certain thresholds within the burnt area algorithm. The information on the land cover was derived from the University of Maryland land cover product described in Section 1.2. The immediate output from the IFI burnt area algorithm is a map indicating the number of times each pixel has been detected as being burnt during the fire season of the year 2000. Consequent post-processing measures are applied to this map to derive the final binary (0/1) map of not burnt/burnt areas.

The large size of the Russian data window made it impossible to process the window altogether. The Russian window covers an area between 4.8^{o} E, 71.4^{o} N and 180^{o} E, 30^{o} N (including the island of Japan and some countries of the Middle East) equating to some 20,000 by 5,000 pixels. Also, given that certain stages of the algorithm needed to be applied to very small blocks of data (200 by 200 pixels) it was important from the start to take into consideration hardware and software resources when planning the processing. The extended Russian window was divided into 9 regions (R1, R2, R3, R4, R5, R6, A1, A2 and A3) as shown in Figure 3. As an indication of the amount of data to be processed the Russian sub-window R3 (see Figure 3) contained 234 (18 times 13) small blocks of data (200 by 200 pixels in size). Each of the large windows overlaps by twenty pixels to reduce the likelihood of cloud contamination at the pixel edge. It is important to note that for the IFI algorithm to

be used, it is necessary that the region of interest to be processed must be of an image size that is a factor of 200. This ensures that the procedures applied to small areas of data are unique. The main fire season in Russia, and therefore the data that was processed, runs from the 1st April until the 31st October 2000 inclusive. After lengthy discussion of preliminary results, it was decided to process the fire season in two stages. This was mainly done to remove commission errors due to phenological changes in the forest causing significant changes in the spectral signal. A summer period was defined that included the months of April to August and an autumn period defined that included the months of September and October.

To control the implementation of the IFI burnt area algorithm, the c-shell script gba_ifi_processor.csh was used (Annex E). This script takes the user through each of the processing stages individually. It is important to note that certain stages should be completed before other stages are started. For example, it is required to pre-process the daily data for the whole fire season before implementing the burnt area detection algorithm. Therefore, it is suggested that each of the stages are implemented separately. Typing gba_ifi_processor.csh at the command line will yield the following instructions:

```
#########################################################
GBA2000 C shell: gba_ifi_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_ifi_processor.csh
<acronym>
<preprocessor flag (0 = no; 1 = yes>
<seasonal algorithm flag (1 = summer; 2 = autumn>
<processor flag (0 = no; 1 = yes>
<postprocessor 1 flag (0 = no; 1 = yes>
<postprocessor 2 flag (0 = no; 1 = yes>
```

where, acronym refers to the sub-window name (e.g. R3), the preprocessor flag activates the pre-processor sub-module, the seasonal algorithm flag indicates the thresholds to be used in the burnt area algorithm, the processor flag activates the burnt area detection sub-module, the postprocessor1 flag activates the first post-processing sub-module and the postprocessor2 flag activates second post-processing sub-module.

Examples are now shown how the user activates the pre-processing module after the data has been read off the archive. This command will apply the same procedure to all dates in the sub-window (e.g. R3), and then process each internal 200 by 200 block of data separately in

memory. To pre-process the data for dates specified in the text file input_dates.txt, the user enters the following at the command line (using sub-window R3 as an example):

gba_ifi_processor.csh R3 1 0 0 0 0

After this has finished (and ensuring that file input_dates.txt has been edited to reflect the whole of the fire season and this data is available) you can apply the IFI burnt area detection algorithm (for the summer period):

gba_ifi_processor.csh R3 0 1 1 0 0

The full text of program gba_ifi_processor.csh can be found in Annex E. Each individual step will now be described in detail.

5.1 IFI data extraction from tape archive

The methods used to extract data from the SPOT Vegetation S1 archive for each sub-window (R1, R2, etc.) is described in Chapter 2. For each of the Russian and Asia (A1, A2 and A3) sub-windows a buffer of twenty pixels is automatically created. This is to reduce the likelihood of any cloud contamination (Section 2.5.1). If a day is missing from the archive you need not be concerned at this stage as long as there is no record of the directory existing (i.e. the directory 20000731 should not exist as this data is not available on the archive). To continue, it is required that land cover information and useful data masks are available for each of the corresponding sub-windows. The programs discussed in Chapter 2, will enable the user to extract land cover information and the masks required for the region of interest. After extraction, the directory should resemble something like this (example shown for window R3):

```
./R3_buffer_umd_lcc
./R3_buffer_umd_lcc.hdr
./R3_buffer_umd_lcc.mask
…
./R3_umd_lcc
./R3_umd_lcc.hdr
…
./R3_umd_lcc_non_forest.mask
./R3_umd_lcc_non_forest.mask.hdr
```

In this directory there are also two text files giving geometrical and geographical information about the buffered and non-buffered sub-window. These files are continuously referred to during the processing and are created during the extraction of the data from tape archive. In the example given they are called:

./R3_buffer_file_info.txt
./R3_file_info.txt

Examples of commands to extract the data from the tape archive are (after editing of the file, input_dates.txt for required dates to extract):

- To extract sub-window R1 without any corresponding land cover information:
  gba_read_vgt_data.csh russia1 0
- To extract sub-window R3 with corresponding land cover information:
  gba_read_vgt_data.csh russia3 1

5.2 IFI pre-processing procedures

The aims of the pre-processing steps, defined by IFI, are to produce daily image products that are:

- Without cloud and snow contaminated pixels.
- Without thin cloud and fire smoke contaminated pixels.
- Without cloud shadow contaminated pixels.
- Without data that have been acquired at extreme viewing zeniths.
- Without pixels affected by MIR saturation.

To produce, what are called here as clean images, programs are used from the collection of generic pre-processing tools described in Chapter 3 and others have been developed separately. For example, one of the programs requires statistics to be calculated of blocks of data 200 by 200 pixels in size (this is the procedure to determine pixels likely to be contaminated by thin cloud or fire smoke). The twenty-pixel buffer around the window is necessary here because of the observed effects of cloud shadow contamination that is not detected because no cloud exists in the image but the cloud shadow does. The final step of the pre-processing module is to extract from the buffered window the non-buffered unique region of interest.

A c-shell script has been written that is called gba_ifi_preprocessor.csh. Typing this program at the command line will yield the following instructions:

```
#######################################################
GBA2000 C shell: gba_ifi_preprocessor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_ifi_preprocessor.csh
<acronym>
<cloud band b0 (grass) threshold (e.g. 500)>
<cloud band mir (grass) threshold (e.g. 500)>
<cloud band b0 (other) threshold (e.g. 200)>
<cloud band mir (other) threshold (e.g. 500)>
<snow band b0 (grass) threshold (e.g. 500)>
<snow band mir (grass) threshold (e.g. 500)>
<snow band b0 (other) threshold (e.g. 200)>
<snow band mir (other) threshold (e.g. 500)>
<viewing zenith angle threshold (e.g. 50.5)>
<cloud height for cloud shadow calculations (e.g. 10000)>
<memory available for cloud shadow processing (e.g. 75)>
<MIR band saturation threshold (e.g. 1000)>
<Thin cloud/fire smoke b0 (grass) threshold (e.g. 191)>
<Thin cloud/fire smoke b0 (other) threshold (e.g. 121)>
<delete intermediate mask (1,0) files (1 = yes, 0 = no)>
<Extract non-buffered data window (1 = yes, 0 = no)>
<Buffer size (pixels)>
```

where, acronym refers to the sub-window name (e.g. R3), the next eight values refer to threshold values provided by IFI to remove cloud and snow from the image, the viewing zenith angle threshold value will remove those pixels acquired at large viewing angles, the cloud height value indicates at what height the clouds are for calculation of cloud shadow and the memory available for cloud shadow processing (see Section 3.4). The MIR band saturation threshold value removes saturated pixels and the thin cloud/fire smoke thresholds are required to effectively remove contaminated pixels. The final three commands are used to indicate whether intermediate mask files are to be kept for reference using the delete intermediate masks flag and to extract from the sub-window the non-buffered unique window that is now free from any cloud shadow contamination.

As is the case with all GBA2000 c-shell scripts, if you type the filename of any c-shell or c program at the command line and press enter, instructions on how to use the program and the specific input parameters to make the program work will be displayed. The full text of the script gba_ifi_preprocessor.csh can be found in Annex E. An example of a command line input to this script is:

gba_ifi_preprocessor.csh R3 500 500 200 500 500 500 200 500 50.5 10000 64 1000 191 121 1 1 20

The first procedure is to mask out clouds and snow. The c program cloud_mask_conditional.c (based on the c program cloud_mask.c described in Section 3.3 and Annex C) was developed to make this processing (see Annex E). Research showed that different thresholds were

required for grassland and non-grassland surfaces. To determine whether the land surface is grassland or otherwise, the UMD land cover product is an input variable to the program. A pixel is classed as being cloud if; band B0's (blue) value is greater than 500, band MIR's (middle-infrared) value is greater than 500 and the land cover is grassland (UMD land cover binary value equals (class) 10), OR band B0's (blue) value is greater than 200, the band MIR's (middle-infrared) value is greater than 500 and the land cover is not grassland (UMD land cover binary value does not equal (class) 10). A pixel is classed as being snow if; band B0's (blue) value is greater than 500, band MIR's (middle-infrared) value is less than 500 and the land cover is grassland (UMD land cover binary value equals (class) 10) OR band B0's (blue) value is greater than 200, the band MIR's (middle-infrared) value is less than 500 and the land cover is not grassland (UMD land cover binary value does not equal (class) 10).

By running the program cloud_mask_conditional.c four times to account for the four different scenarios, joining together cloud masks and snow masks for both land covers and then multiplying snow and cloud masks together, a composite cloud and snow mask is produced. The adding together and multiplication of mask files is undertaken using the generic c programs join_mask_files.c and apply_mask.c (Annex A). An example of the programs is now given (refer to Annex E for a full description of this program):

cloud_mask_conditional $pixels $lines
$year$month$day/$acronym_$year$month$day_grass_cloud
$acronym_buffer_umd_lcc 10 1
$year$month$day/$acronym_$year$month$day_b0
$cloud_b0_grass
$year$month$day/$acronym_$year$month$day_mir
$cloud_mir_grass
4 1

The example shown above produces a binary (0/1) file containing those pixels flagged as being cloudy for pixels defined as being grass. After determining the other situations, snow and cloud images for both grassland and non-grassland surfaces are added together, as in this example for cloud:

join_mask_files
$year$month$day/$acronym_$year$month$day_grass_cloud
$year$month$day/$acronym_$year$month$day_other_cloud
$year$month$day/$acronym_$year$month$day_cloud
$pixels $lines

The cloud and snow masks are multiplied together to produce the joint cloud and snow mask:

apply_mask
$year$month$day/$acronym_$year$month$day_cloud
$year$month$day/$acronym_$year$month$day_snow
$year$month$day/$acronym_$year$month$day_cloud_and_snow $pixels $lines 1

The next step is to remove those data that have been acquired at extreme viewing zeniths. The threshold provided by IFI was to remove all those pixels with a viewing zenith angle value greater than 50.5 degrees. This binary mask was produced using the c program view_zenith_mask.c (see Section 3.1 and Annex C). The viewing zenith binary mask was then applied to the cloud and snow mask derived above (using the generic c program apply_mask.c) to produce an updated version of the cloud and snow mask without any pixels acquired at extreme viewing angles.

The next step was to mask out any remaining pixels that might be thin cloud or fire smoke. To do this an index value (presented here as $I_{B2/B0}$) of band B2's (red) DN divided by band B0's (blue) DN was calculated for each remaining pixel. The mean value of the remaining pixels for each block of 200 by 200 pixels of index $I_{B2/B0}$ is then calculated. Those pixels that have index values greater than the mean value are then excluded from further analysis. The pixels that are excluded are clean data and kept as value 1 in the thin cloud/fire smoke mask. The standard deviation of those pixels with index values less than the mean value is then calculated ($SD(I_{B2/B0})$). A pixel is determined as being clean if it satisfies the following conditions; if $I_{B2/B0}$ is greater than $1 + 0.75 * SD(I_{B2/B0})$ and band B0's value is less than or equal to 190 and the land cover is grassland (UMD land cover binary value equals (class) 10), OR if $I_{B2/B0}$ is greater than $1 + 0.75 * SD(I_{B2/B0})$ and band B0's value is less than or equal to 120 and the land cover is not grassland (UMD land cover binary value does not equal (class) 10). If the situation is otherwise, the pixel is considered to be either thin cloud or cloud smoke and is flagged as so in the output mask product. The c program ifi_fire_smoke_mask.c makes these calculations for each block of data and produces a binary (0/1) mask indicating the presence of thin clouds or fire smoke. An example of the input commands and variables to this program is given (see Annex E):

ifi_fire_smoke_mask $pixels $lines
$year$month$day/$acronym_$year$month$day_b0
$year$month$day/$acronym_$year$month$day_b2
$year$month$day/$acronym_$year$month$day_cloud_and_snow
$acronym_buffer_umd_lcc
$year$month$day/$acronym_$year$month$day_fire_smoke
10
$b0_grass_threshold $b0_other_threshold

The next step is to dilate the cloud, snow and fire smoke masks so that pixels next to the contaminated pixels are removed from the data. A small number of dilation algorithms were examined for this task, including a FOCAL MAXIMUM function available with the ERDAS

IMAGINE image processing software. However, it was decided to use a more simple approach and remove just a single layer of pixels from around each of the contaminated pixels. The generic c program erode_mask.c, (Annex A) was used for this task. The program was applied separately to the cloud mask, the snow mask and the thin cloud/fire smoke mask with the viewing zenith angle mask being used as a reference for all of those pixels that are considered as being erodable. An example of the input commands and variables to this program is given (see Annex A):

erode_mask
$year$month$day/$acronym_$year$month$day_cloud
$year$month$day/$acronym_$year$month$day_vza_mask
$year$month$day/$acronym_$year$month$day_cloud_buffer
$pixels $lines

The next step is to calculate those pixels that could possibly be contaminated by cloud shadow given the presence of clouds and fire smoke in the scene. The method provided by IFI separated the circle surrounding the pixel into eight parts, corresponding to 45 degrees for each part, and defined a matrix in a window of nine by nine pixels. Depending on the sun position, cloud shadow masks (indicated by values in the matrix) would be applied in a direction opposite to the Sun's azimuth angle, where the cloud shadow would be situated. The limitation of this would be that sometimes the cloud shadow would extend beyond the limit of the nine by nine pixel matrix (as observed) and fail to capture all of the cloud shadow. After consultation it was decided to use the c program shadow_mask.c that had already been developed and tested for this purpose. Prior to the application of this program it was necessary to multiply together the dilated cloud mask and the thin cloud/fire smoke mask, as both of these products were considered capable of casting shadows. The generic c program apply_mask.c was used to do this. The c program shadow_mask.c was applied using a reference cloud height of 10,000m. It was acknowledged that this cloud height was an overestimate in most cases, especially for pixels detected as fire smoke but for consistency and the need to be certain that contaminated pixels are removed this value was agreed upon. An example of the input commands and variables to this program are given (see Annex C):

shadow_mask
$year$month$day/$acronym_$year$month$day_all_cloud_buffer
$year$month$day/$acronym_$year$month$day_vza_mask
$year$month$day/$acronym_$year$month$day_shadow
$year$month$day/$acronym_$year$month$day_vza
$year$month$day/$acronym_$year$month$day_sza
$year$month$day/$acronym_$year$month$day_vaa
$year$month$day/$acronym_$year$month$day_saa
$pixels $lines $start_lon $start_lat $cloud_height $memory

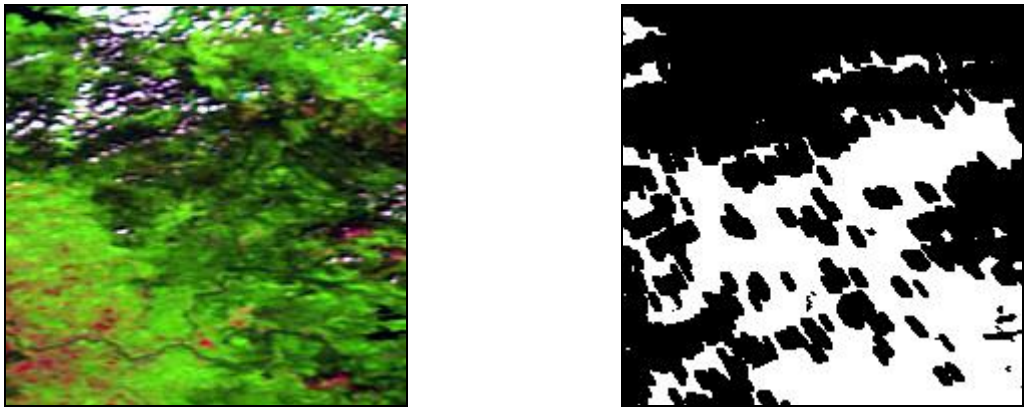
Once the shadow mask was produced, the next step was to check for, and remove, pixels affected by saturation in the middle-infrared (MIR) channel. IFI suggested that the SPOT Vegetation S1 status product provided in the data archive was used for this purpose. However, after private communications with colleagues and users of the data, it was decided that this product was unreliable. A more simple approach was adopted, that provided satisfactory results. A user-defined threshold would be applied to the MIR channel and the two adjacent pixels to the left and right of the offending pixel would also be removed. This latter step was implemented after observations that the saturated line would not only affect the value of the adjacent pixel but as the pixel next but one away. After testing, all major saturation artifacts were successfully removed. A MIR threshold value of 1000 was selected. The c program mir_sat_mask.c (see Section 3.2 and Annex C) was developed to remove the saturated (and adjacent) pixels. An example of the input commands and variables to this program is given (see Annex C):

mir_sat_mask
$year$month$day/$acronym_$year$month$day_mir
$year$month$day/$acronym_$year$month$day_mir_out
$pixels $lines $mir_threshold

The next step involves joining together the mask files to create the contaminated pixel image that will be used in the next stage of processing. The important masks that need to be joined are the cloud shadow mask, the MIR saturation mask, the dilated snow mask and the non-burnable surface land cover mask produced when extracting the data from archive. The generic c program apply_mask.c is used to do this. A contaminated pixel mask exists within each date directory (e.g. 20000401, 20000402 etc.) of each sub-window (e.g. R3). It is now possible to remove all of the intermediate mask products by flagging the appropriate input parameter in the c-shell script gba_ifi_preprocessor.csh.

Until now, each of the daily datasets has an overlap region of twenty pixels surrounding the unique region of interest. Now that the final contaminated pixel product is available this buffer can be removed. By utilizing the generic c program snip.c (Annex A), the data are overwritten by the non-buffered data (composing of four bands of spectral data, four bands of angle data and the contaminated pixel mask). It is important here not to delete any text files (especially those containing geometrical information) or any UMD land cover product, as these data will be referred to in subsequent processing stages. An example of the resultant contaminated pixel mask is shown in Figure 12. The image on the left (a) is the original data displayed as an RGB image (Red = band MIR, Green = band B3, Blue = band B2). The date

of the acquisition is the 3rd July 2000 for a small region in central Siberia. The image on the right is the contaminated pixel mask that is used in the next stage of the processing. The areas shown in white indicate good quality data to be considered for burnt area mapping. The regions shown in black are those areas that will not be considered for burnt area mapping.



(a) RGB Colour Composite (b) Contaminated Pixel Mask

Figure 12: Examples of original SPOT Vegetation S1 data (a), and the resultant contaminated pixel mask (b), produced by the IFI pre-processing algorithm, for a small region in central Siberia. The image was acquired on the 3rd July 2000. The image shown is 200 by 200 pixels in size.

5.3 IFI burnt area algorithm procedure

The burnt area detection algorithm developed by IFI was originally coded in ERDAS Imagine's Modeller software. It was decided from an early stage to rewrite the software in IDL in keeping with other algorithms that were coded in this way. Around the IDL program, a c-shell script was developed to automate the running of the program. Although the algorithm is applied to the sub-window, some of the algorithm's stages were applied to small regions of interest 200 by 200 pixels in size. The c-shell script gba_ifi_ba_algorithm.csh was written to automate the implementation of the burnt area algorithm. Typing gba_ifi_ba_algorithm.csh at the command line will yield the following (Annex E):

```
########################################################
GBA2000 C shell: gba_ifi_ba_algorithm.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
########################################################
Syntax: gba_ifi_ba_algorithm.csh
<acronym>
<path to gba_ifi_ba_algorithm.pro (IDL composite program)>
<path to IDL executable binary>
<seasonal algorithm flag (1 = summer, 2 = autumn)>
```

where, acronym refers to the sub-window name (e.g. R3), path to gba_ifi_ba_algorithm.pro is the directory path to where the IDL programs are stored and path to IDL executable binary is the full directory path to where IDL is launched (type which IDL at the command line to find this out). The flag for the seasonal algorithm to be implemented is also given at the command line.

Before launching this c-shell script ensure that all the daily data (with contaminated pixel mask file) are available for the summer or autumn time period. Edit the file input_dates.txt to reflect your choice of season (e.g. set to process the summer time period). After initiating, for each daily dataset to be processed BSQ files containing all of the necessary input files are created in a directory located within the working directory as shown here (e.g. R3):

./R3/R3_ifi_ba/R3_20000401_bsq

The BSQ files which are created using the SUN OS command cat are composed of the files that are required to detect burnt areas, bands B3 (near-infrared) and MIR (middle-infrared), four angle bands (viewing zenith, viewing azimuth, sun zenith and sun azimuth) and the contaminated pixel mask. The next stage of the processing creates a list of all the daily images that are available for processing. If days are missing then please make a note of these as although the program will function with dates missing you will need to be aware of these during a later stage of processing. The IDL program gba_ifi_ba_algorithm.pro is copied into the current directory, IDL is started and the algorithm is run.

The full text of the IDL program gba_ifi_ba_algorithm.pro is available in Annex E. The first step in determining the burnt areas is the creation of an index using band B3 (near-infrared) of the daily S1 product ($S1_{B3}$) and an intermediate composite (IC) product of band B3 (near-infrared) that is continually updated on a daily basis with non-contaminated data (IC_{B3}). The index (I_{B3B3}) is a result of dividing the difference between the $S1_{B3}$ pixel value and the IC_{B3} pixel value, by the sum of the $S1_{B3}$ pixel value and the IC_{B3} pixel value. In this case, it is necessary to exclude from the calculations those occurrences where the $S1_{B3}$ pixel value equals zero or when the IC_{B3} pixel value equals zero. If these conditions are not satisfied then the I_{B3B3} value for that particular pixel is not calculated. This indicates that for the first date that is processed, there is no I_{B3B3} array calculated because even though S1 data is available there is no IC_{B3} composite to derive the index from. For the first day in the sequence, the following step will then be ignored (i.e. no burnt area map on day one can be produced).

Given that an I_{B3B3} image is available, the mean ($M(I_{B3B3})$) and standard deviation ($SD(I_{B3B3})$) of values in the index are calculated, ignoring those values masked out in the contaminated pixel binary and those that are zero in the index array. The criteria that a pixel is a potential burnt pixel is valid if these conditions are satisfied; if I_{B3B3} is less than ($M(I_{B3B3}) - 2 * SD(I_{B3B3})$) AND band B3 (near-infrared) is less than 260 AND band MIR (middle-infrared) is greater than 250.

The creation and continual updating of the IC_{B3} array occurs after it is determined whether any potential burnt pixels occur in the image for that particular day. After any changes to the IC_{B3} occur, this array is then kept in memory for generation of the index product (I_{B3B3}) the following day. This process essentially fills gaps with non-contaminated pixels and also averages out spectral signatures for both burnt and non-burnt vegetated surfaces. To create the IC_{B3} image the first step is to calculate the phase angle, termed here PA. The formula to calculate the phase angle is presented in Annex E. All of the values in the image product are then replaced with a zero if the contaminated pixel mask for that date is also a zero. It is at this stage in the algorithm that differences occur between the summer and autumn datasets. The threshold values used to determine if the composite image is modified are more relaxed for the autumn period. The value of the IC_{B3} composite image is calculated by the following formula for the summer period (April to August inclusive):

- If IC_{B3} does not equal 0 AND ($S1_{B3}$ does not equal 0 AND the viewing zenith angle is less than or equal to 70 (in the original 8-bit scale) AND the phase angle is less than 0.8); then the new value of IC_{B3} equals ($IC_{B3} + S1_{B3}$) / 2.
- If (IC_{B3} equals 0 AND ($S1_{B3}$ does not equal 0 AND the viewing zenith angle is less than or equal to 70 (in the original 8-bit scale) AND the phase angle is less than 0.8)) OR (IC_{B3} does not equal 0 AND $S1_{B3}$ equals 0); then the new value of IC_{B3} equals $IC_{B3} + S1_{B3}$.
- Otherwise the new value of IC_{B3} remains unchanged (i.e. IC_{B3} equals IC_{B3}).

The value of the IC_{B3} composite image is calculated by the following formula for the autumn period (September and October):

- If IC_{B3} does not equal 0 AND ($S1_{B3}$ does not equal 0 AND the viewing zenith angle is less than or equal to 90 (in the original 8-bit scale) AND the phase angle is less than 1.0); then the new value of IC_{B3} equals ($IC_{B3} + S1_{B3}$) / 2.

- If (IC_{B3} equals 0 AND ($S1_{B3}$ does not equal 0 AND the viewing zenith angle is less than or equal to 90 (in the original 8-bit scale) AND the phase angle is less than 1.0)) OR (IC_{B3} does not equal 0 AND $S1_{B3}$ equals 0); then the new value of IC_{B3} equals $IC_{B3} + S1_{B3}$.
- Otherwise the new value of IC_{B3} remains unchanged (i.e. IC_{B3} equals IC_{B3}).

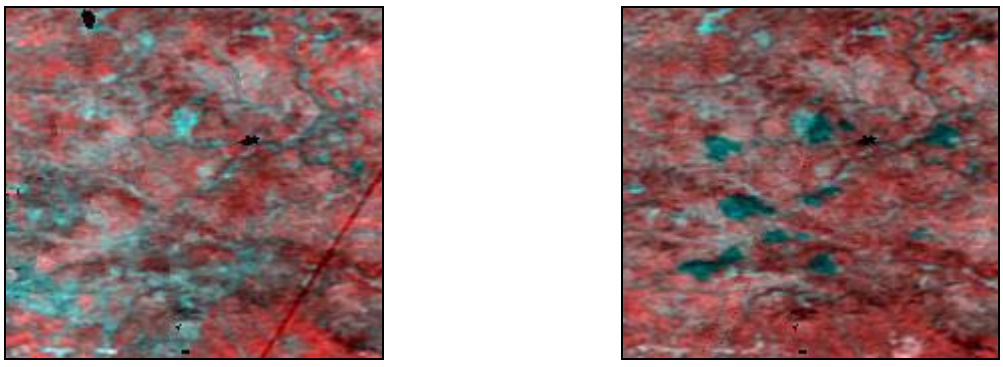
The same rules are applied to generate an intermediate composite image of the middle-infrared channel (IC_{MIR}). Although this product is not directly used in determining the potentially burnt pixels, it is written out to file every month and used in the post-processing algorithms.

The values of the IC_{B3} image are rounded up and kept in memory for use in determining the burnt areas in the next day's processing. The values in the intermediate composite image are continually being averaged out if there are clean data available. If there is a change in the spectral signature caused by a fire event, then this change filters through into the intermediate composite value where it is again continually adjusted to represent the new spectral conditions of the land surface. This results in pixels that have been affected by fire having more than one detection as being potentially burnt throughout the fire season. This information is then used to refine the map of actual burnt pixels in the post-processing stages.

The output files from the IDL program gba_ifi_ba_algorithm.pro (after running the script twice, once for the summer and once for the autumn period) include a potential burnt area image for each day in the fire season ordered consecutively from the first date (e.g. 20000401 = 01, 20000402 = 02 etc.) to the final date (20001031 = 212). It is important here to note that any days missing will be absent from the consecutive numbering scheme. In the case of the SPOT Vegetation year 2000 dataset, dates 20000731 and 20001006 are absent. At the end of every month the intermediate composite images (IC_{B3} and IC_{MIR}), in floating point format, are also written to file. These files are used to assess the results, determine post-processing procedures and used as reference images. After the processing is complete, the following files should be present within each sub-directory (example given of window R3):

```
./R3/R3_ifi_ba/R3_ifi_pba_1
./R3/R3_ifi_ba/R3_ifi_pba_2
…
./R3/R3_ifi_ba/R3_ifi_pba_211
./R3/R3_ifi_ba/R3_ifi_pba_212
./R3/R3_ifi_ba/R3_ifi_ic_b3_30
./R3/R3_ifi_ba/R3_ifi_ic_b3_61
…
./R3/R3_ifi_ba/R3_ifi_ic_b3_212
./R3/R3_ifi_ba/R3_ifi_ic_mir_30
./R3/R3_ifi_ba/R3_ifi_ic_mir_61
./R3/R3_ifi_ba/R3_ifi_ic_mir_212
```

Header files are written (that can be read directly into ENVI) for the output files and the BSQ files are deleted from the disk. The burnt area algorithm implementation is complete. Figure 13 shows the construction of the intermediate composite image at day 91 (end of June) and at day 152 (end of August) of a region in central Siberia. The composites are displayed as RBG images (Red = band IC_{B3}, Green = band IC_{MIR}, Blue = band IC_{MIR}). New fire scars are observed in the August IC product in dark blue colours (indicating a drop in the near-infrared spectral signal).



(a) Composite after 91 days (b) Composite after 152 days

Figure 13: Example of intermediate composite images for a small region in central Siberia. The image on the left (a) is the composite product at the end of June 2000, the image on the right (b) is the composite product at the end of August 2000. The image shown is approximately 180 by 175 pixels in size.

5.4 IFI post-processing (stage 1) procedure

The first stage of post-processing involves summing the number of indications that each pixel has potentially burnt for each of the time periods (summer and autumn). This yields, for each pixel, the total number of times the pixel has satisfied the IFI criteria for burnt area detection over each season (in the case of Russia this is 152 days for the summer period and 60 days for the autumn period, taking into account missing data). A c-shell script is available that

does this automatically called gba_ifi_postprocessor1.csh (see Annex E). The input variables are indicated when you enter gba_ifi_postprocessor1.csh at the command line:

```
#######################################################
GBA2000 C shell: gba_ifi_postprocessor1.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_ifi_postprocessor1.csh
<acronym>
<seasonal algorithm flag (1 = summer, 2 = autumn>
```

where, acronym refers to the sub-directory block name (e.g. R3) and the seasonal algorithm flag refers to the time period of interest.

The first step in the post-processing is actually undertaken in the c-shell script gba_ifi_processor.csh. A byte array containing only zeros is created using the generic c program make_raster.c (Annex A) in the current directory. All other potential burnt area image files are added to this image. Each individual potential burnt area image are added together so that, for each pixel, the final pixel value for the output product will be a sum of all values over the fire season. The adding of consecutive files is undertaken using the generic c program add_files.c (Annex A). An example is now shown:

```
add_files
$acronym_ifi_pba $acronym_ifi_pba_122 temp_ifi_ba_image
$pixels $lines 1
```

Another useful piece of information is the day that the pixel is first detected as being potentially burnt. If the potentially burnt pixel is confirmed later to be an actual burnt pixel then the information on the date of burn can be used to compile monthly statistics. This calculation is made using the c program ifi_day_of_burn.c (see Annex E). As the program loops through the potential burnt pixel maps, the program continuously updates an array containing the date of first detection, expressed in a byte scale between 1 (1st of April 2000) through to 212 (31st October 2001). Again, please note here that the actual number of days between the two dates is 214, but because of missing data on the 31st of July and the 6th of October then this total number is reduced to 212. This fact needs to also be remembered for determining which day belongs to which month. An example is now shown of the c program:

ifi_day_of_burn
$acronym_ifi_pba $acronym_empty_image $acronym_day_of_burn
$pixels $lines 2

The final step is to compress the potential burnt image files and construct file headers for the images. After the c-shell script has finished for both the summer and autumn seasons, the directory structure should resemble the following example (ignoring header files):

./R3 /R3 _ifi_ba/R3_ifi_pba_t1
./R3 /R3 _ifi_ba/R3_ifi_pba_t2
./R3 /R3 _ifi_ba/R3_ifi_pba_1.gz
./R3 /R3 _ifi_ba/R3 _ifi_pba_2.gz
…
./R3 /R3 _ifi_ba/R3 _ifi_pba_212.gz
./R3 /R3 _ifi_ba/R3 _day_of_burn_t1
./R3 /R3 _ifi_ba/R3 _day_of_burn_t2

Figure 14 shows an example of the summed potential burnt area image for a region in central Siberia (identical to that shown in Figure 13). The values within the image range from zero (black) through a linear grey scale to seven detections (white)

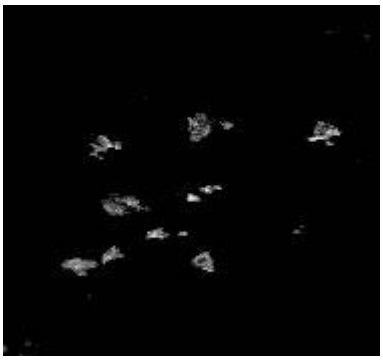


Figure 14: Example of a summed potential burnt area image for a small region in central Siberia. The image is expressed as a linear grey scale between no detections (black) and seven detections (white). The image shown is 180 by 175 pixels in size.

5.5 IFI post-processing (stage 2) procedure

To determine those pixels, or areas, that actually burnt during the year 2000 the data requires further post-processing. Algorithms have been developed to maximise the removal of false detections while ensuring that those pixels that are actually burnt are preserved in the final burnt area maps. Based on the analysis of intermediate composite images and the potential burnt area images the following method was proposed:

- The exclusion of all burnt pixels situated within land cover class 9 of the University of Maryland (UMD) land cover product. This corresponds to a land cover type of

open shrubland. This class was excluded because all observed potential burnt area pixels were false detections.

- The dataset is analysed on a monthly basis with reference made to the intermediate composite product available at the end of each month. Referring to the detected day of burn product indicates the month of burning.
- Potential burnt pixels would be excluded that were located in a one pixel expanded water mask, generated from the UMD product.
- For each summer month (April through to August inclusive), potential burnt area pixels are excluded if they are located in spectral space outside of ± 2 standard deviations of the range of values in bands B3 and MIR. This analysis is made for each vegetated land cover class of the UMD product.
- For autumn months (September and October), potential burnt area pixels are excluded if they are situated at a location greater than -1 standard deviation of the NIR histogram composed of those pixels outside of the potential burnt area mask. Again, the analysis is conducted for each vegetated land cover class of the UMD product.
- All isolated pixels, and those making up clumps of less than 3 pixels, are excluded from the potential burnt area product.
- A buffer zone is created around each remaining clump of potential burnt area pixels using a 5x5 window. For each class of the UMD product, mean values for the spectral bands B3 and MIR were calculated for those pixels not included in the updated potential burnt area mask and included in the updated potential burnt area mask. Distance values are then calculated for each pixel in this buffer zone for the mean values of the burnt area and non-burnt area masks. The distance values are compared to each other, thus determining the class association of the pixel located in the buffer zone. Those pixels still remaining are classed as burnt areas in the final product.
- This analysis is undertaken at the large window scale (e.g. R1, R2, etc.).

Based on the burnt area results obtained from this second post-processing stage certain restrictions were placed on these algorithms for different land cover types and regions. In particular, the 5x5 window was found to over-exaggerate the size of the burnt area, especially outside forest regions. As a compromise, a 3x3 window expansion was used in forested regions (forest area was derived from the UMD global land cover product) and no expansion of the burnt area was undertaken outside of forested regions. In addition, with the exclusion

of windows R1 to R6, A1, A2 and A3, no removal of single pixels was undertaken, including windows EU, AR, A4 and A5. Detected burnt area pixels associated with class 9 (open shrubland) of the UMD land cover product were only removed from windows R1 to R5 (corresponding to high northern latitudes). A summary of the post-processing procedures applied to each window is given in Table 4 located in Chapter 13 of this report.

The c-shell script gba_ifi_processor.csh activates the script gba_ifi_postprocessor2.csh (Annex E) that has been developed to implement the criteria set out above. The input variables required by this program are:

```
#########################################################
GBA2000 C shell: gba_ifi_postprocessor2.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_ifi_postprocessor2.csh
<acronym>
<eroded water mask mask interations (1 or 2)>
<path to GBA2000 IDL source programs>
<path to IDL executable binary>
```

where, acronym refers to the sub-window (e.g. R3), the number of iterations to apply to the water mask dilation (one or two pixel dilations are supported) and the full paths to the IDL algorithm programs and IDL executable binary.

The first stage of the algorithm applies a dilated water mask to the summed potential burnt area images and the images indicating the estimated first day of burning. The user here has a choice of applying a one or two pixel dilated water mask to the potential burnt area products. For the Russian windows a one pixel dilated water mask was applied. The programs used to do this are the generic c programs erode_mask.c and apply_mask.c (see Annex A). An example of the input commands and variables to the generic c program erode_mask.c in this case is:

```
erode_mask
$acronym_umd_lcc.mask $acronym_full_image
$acronym_umd_lcc_erode.mask
$pixels $lines
```

The summed burnt area product is then separated into monthly products as indicated by the value of the day of burn image. A c program called ifi_split_months.c (Annex E) has been written to make this separation. The input commands and variables to this program are shown as an example for the month of May:

ifi_split_months
$acronym_ifi_pba_erode_t1
$acronym_day_of_burn_erode_t1
$acronym_ifi_pba_05
$pixels $lines
31 61

where, the two input files derived from the first post-processing stage are required, followed by the name of the output file and geometry of the files and the dates between which (and inclusive of the two values) all those values within the day of burn image should be considered as commencing in May.

The processing is undertaken for each season, summer and autumn, by changing the date range at the command line. The next step is the removal of a specific land cover type open shrubland. This is done by creating a land cover mask that consists solely of this class (class nine in the UMD product) by using the c program land cover_mask_ifi.c (see Annex E) with the following input commands:

land cover_mask_ifi
../$acronym_umd_lcc
$acronym_umd_class9_mask
$pixels $lines

The mask is then applied to each of the newly created monthly, potential burnt area products using the generic c program apply_mask.c (Annex A).

After some renaming of the intermediate composite images, the next stage of the algorithm is to remove false detections. For each of the summer months (April to August inclusive), we calculate mean and standard deviation values for each class of the UMD product for those pixels belonging to the potential burnt area images in the spectral bands B3 and MIR. For both bands an area has been defined (minimum/maximum limits) for the potential burnt areas. If those pixels indicated as potentially burnt are located outside of ± 2 standard deviations then they are excluded from further analysis and are not considered as potentially burnt. This applies to both B3 and MIR channels. For autumn months (September and October), we calculate mean and standard deviation values for each class of the UMD map for pixels not belonging to the potential burnt area mask in the B3 band only. We suppose that true burnt areas should have a low value in the near-infrared (B3), less than those pixels that are not burnt and, therefore, will be located in the region of the mean minus one standard deviation for each UMD class. However, this is not always the case. For the sub-windows of A2 and A3 in central Asia (see Figure 3), the value that splits the region of potentially burnt

pixels (mean minus a factor multiplied by the standard deviation) was changed. When using a factor of 1.0 (standard deviations) on these windows, all potential burnt area pixels were removed. Optimised values of 0.2 standard deviations for the autumn months of the window A2 and 0.5 for the month of September for the window A3 were derived from analysis of the burnt area maps with single day imagery. The value of this factor is set at the beginning of the c-shell script gba_ifi_postprocessor2.csh and read by the program that processes just the autumn months. Two c programs have been written to make these tests on the data, one applied to each summer month called ifi_clean_pba.c and one applied to the autumn months called ifi_clean_pba_autumn.c. Both are presented in Annex E. The inputs to the two programs are as follows:

ifi_clean_pba
$acronym_ifi_pba_$month
$acronym_ifi_ic_b3_$month
$acronym_ifi_ic_mir_$month
../$acronym_umd_lcc
$pixels $lines
$acronym_ifi_pba_$month_out

ifi_clean_pba_autumn
$acronym_ifi_pba_$month
$acronym_ifi_ic_b3_$month
../$acronym_umd_lcc
$pixels $lines
1.0
$acronym_ifi_pba_$month_out

The next step is to remove single pixels, and those in clumps of less than three pixels that are indicated as being burnt from the dataset. To determine a number of pixels of contiguous groups, we applied the function of an eight-pixel neighbour clump dilation. From the clump image, pixels are sieved if located in clumps of less than three pixels. AN IDL program called gba_ifi_sieve_class.pro is used, controlled automatically by the c-shell script. Details of this IDL program are given in Annex E. The resultant products contain only those clumps of greater than or equal to three pixels.

After analysis of burnt area pixels still present after previous processing steps with single day imagery, it was observed that an under estimation of the area burnt was being made by the algorithm at this stage. To correct this problem, a buffer zone was created around each pixel using a window size of five by five pixels and a focal maximum procedure. This procedure replaces the value of the centre pixel with the maximum value of the surrounding pixels in the kernel (which can either be a zero or a one). A c program called ifi_focmax.c (see Annex E) is used to create this buffer zone with the following input variables:

ifi_focmax
$acronym_ifi_pba_$month_sieve
$acronym_ifi_ic_b3_$month_sieve_buffer
$pixels $lines

where, the input file is the sieved burnt area image (after clumping) for each month. A buffer zone is created around each monthly product. The final stage of this post-processing step is to ensure the suitability of each pixel in the buffer zone actually belongs to the burnt area class. To do this, for each land cover class of the UMD product, mean values of those pixels belonging to and not belonging to the potential burnt area product for bands B3 and MIR are calculated. For each pixel located in the buffer zone, a distance estimator is then calculated between the mean of the potential burnt pixels and those not burnt. In the case of the distance between the spectral values of the pixel located in the buffer zone and the mean of all potential burnt area pixels, the calculation made is:

$$D1_{(i,j)} = [(\mu IC'_{B3(i,j)} - B_{B3(j)})^2 + (\mu IC'_{mir(i)} - B_{mir(j)})^2]^{1/2}$$

where, $\mu IC'_{B3(i,j)}$ is the mean value of each UMD class (i) inside the potential burnt area mask for the band B3 (and similar for the MIR band) and $B_{nir(j)}$ is the value of the buffer pixel for the band B3 (and similar for the MIR band). A similar distance $D2_{(i,j)}$ is created for the pixels outside of the potential burnt area mask.

The two distances for each pixel are compared to each other to determine the class ownership of each pixel in the buffer zone. A factor is applied to the distance between the buffer pixel and the pixels that are not burnt. For the summer months (April to August inclusive) this factor is 0.7 and for the autumn months (September and October) this factor is 0.8. Hence, if the distance $D1_{(i,j)} < 0.7 * D2_{(i,j)}$ for the summer months (replace with 0.8 for the autumn months) then the pixel in the buffer zone is closer to the spectral characteristics of the burnt area pixels and is identified as being burnt in the final product. If this condition is not satisfied then the pixel is identified as not being burnt.

A c program called ifi_region_grow.c (see Annex E) is available to make this calculation for each pixel in the buffer zone. The input commands to this program are:

ifi_region_grow
$acronym_ifi_pba_$month_sieve
$acronym_ifi_pba_$month_sieve_buffer
$acronym_ifi_ic_b3_$month
$acronym_ifi_ic_mir_$month
../$acronym_umd_lcc
$pixels $lines
final_ba_images/$acronym_ifi_ba_$month 1

where, the inputs to the program are the buffered potential burnt area image, the non-buffered burnt area image, the intermediate composite images and the UMD land cover product. The final input parameter (here shown as a one) indicates that the month being processed is located in the summer period (0.7 factor) as opposed to the autumn period (where a value of two is entered here).

6 UTL algorithm implementation module

The Portuguese GBA2000 partner, comprising of João Silva and José Pereira, located at the Universidade Técnica de Lisboa (UTL) in Portugal, were responsible for producing two algorithms for the detection of burnt areas from SPOT Vegetation data. The algorithms were originally based on the same method, which was to use Fisher's Linear Discriminant analysis methods to detect burnt areas by multi-temporal change detection methods applied to a number of parameters derived from the image data. The algorithms differ in the parameters used to make the discrimination between burnt and not burnt land surfaces and the values of thresholds used. Each algorithm is now described in detail. The algorithms were developed, calibrated and validated over two geographical regions, south-east Africa (Africa 1) and the Iberian Peninsula. A third algorithm (based on a modified version of the Africa 1 algorithm) was developed by the author of this report and used in parts of northern Asia. After the release of the beta version of the GBA2000 product, it was discovered that the original UTL burnt area algorithm for Africa was making a significant underestimation of the true burnt area consistently over the sub-Saharan continent. An additional burnt area algorithm (Africa 2) was developed by means of a supervised classification technique based on the Classification and Regression Trees (CART) theory (Breiman *et al*., 1984) and applied to the whole of the sub-Saharan continent. This algorithm is referred to as the UTL Africa 2 algorithm and is described in Section 6.4. The UTL Africa 1 is applied operationally over the A1 window (Figure 3).

6.1 UTL Africa 1 module

An algorithm for burnt area detection in the savanna grassland and woodlands of southern Africa was developed by colleagues at UTL, Portugal (Silva *et al*. 2002a). The algorithm was developed using SPOT Vegetation imagery covering the area shown in Figure 15. The algorithm was validated using Landsat TM imagery (the coverage of this scene is also shown in Figure 15). The algorithm consists of three parts:

- Compositing of daily SPOT Vegetation images using a minimum near-infrared criterion over a monthly time period.
- Applying the burnt area detection algorithm based on linear discriminant analysis methods.
- Applying post-processing procedures to improve the accuracy of the results.

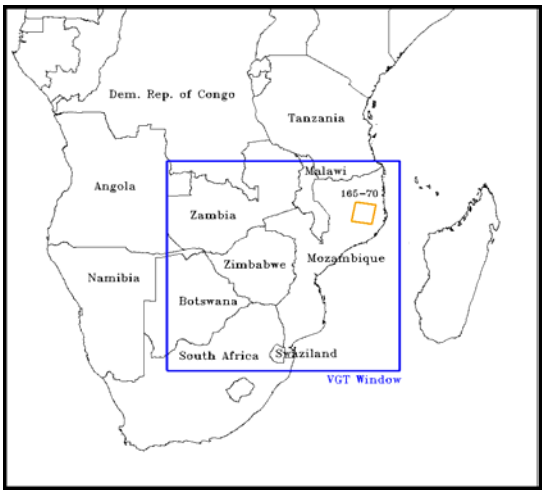


Figure 15: The Africa data window used by colleagues at UTL to develop the burnt area algorithm. The orange window indicates the location of the Landsat TM image that was used for validating the algorithm.

The extraction of data is undertaken using standard procedures described in Chapter 2. The algorithm utilises pre-processed (Chapter 3), monthly composites of data that have been created using a minimum near-infrared criteria (Section 4.1). The algorithm was applied to the southern Africa window (SA), the tropical Africa window (TA) and the central Africa window (CA) and also over the A1 window. A complete description of the algorithm specific pre-processing, compositing, burnt area processing and post-processing stages are now given. The burned area results derived for the African continent were replaced by a burnt area algorithm based on CART theory (UTL Africa 2 algorithm) that is described in Section 6.4.

6.1.1 UTL pre-processing procedure for all UTL algorithms

The pre-processing requirements of the UTL burnt area algorithm for all of the calibration sites are the effective removal of cloud and cloud shadow, non-vegetated surfaces, saturated pixels in the MIR band and pixels acquired at extreme viewing zeniths from daily SPOT Vegetation datasets. The c-shell script gba_preprocessor.csh is used (Chapter 3) because all of the above requirements can be satisfied. The thresholds used for the detection of pixels that are cloudy are 180 for the B0 band and 180 for the MIR band. The cloud mask was also dilated by one pixel before calculation of the cloud shadow mask. An example of the command used to pre-process the data for all of those dates indicated in the text file input_dates.txt, is:

gba_preprocessor.csh SA 1 60 1 1000 1 4 1 180 180 1 1 10000 75 1 all 1

For each day, the pre-processed data would contain data that were masked for cloud, cloud shadow (assuming a cloud height of 10 km), saturation in the MIR channel (all pixels with a DN > 1000), non-vegetated and water surfaces and pixel data acquired at extreme viewing zeniths (> 60 degrees). The threshold values for the cloud masking were extensively tested to effectively remove cloudy pixels but also to preserve those pixels that represented bright, non-cloudy pixels on the Earth's surface.

6.1.2 UTL compositing procedure for all UTL algorithms

The pre-processed SPOT Vegetation data was composited over a time period of one month according to a minimum near-infrared (minNIR) criteria (Section 4.1). The text file, input_dates.txt was edited to reflect the time period to be composited. To make the composites, the c-shell script gba_min_nir_composite.csh was used. An example of the command used is:

gba_min_nir_composite.csh SA ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin all m 1

This command indicates that level two products (including for each pixel the number of cloud free days, the day in the time period from which the pixel was selected for the composite and the status image) were to be produced from fully pre-processed data over a monthly time period.

6.1.3 UTL burnt area algorithm procedure for Africa 1

For the detection of burnt areas with the UTL algorithm for Africa, six variables need to be calculated. Three are related to temporal differences (pre-fire conditions compared to post-fire conditions) and three are related only to post-fire conditions. They are described below, where t is the monthly composite being processed for burnt areas, and t-1 is the previous month's composite:

- Variable 1 (var1): The value of the composited pixel in the near-infrared channel (B3) at time t subtracted from the value of the composited pixel in the near-infrared channel (B3) at time t-1.
- Variable 2 (var2): The value of the composited pixel in the near-infrared channel (B3) at time t.
- Variable 3 (var3): The value of the albedo of the composited pixel (albedo is defined here as the average of the composited pixel in the red (B2), near-infrared (B3) and

middle-infrared (MIR) channels) at time t subtracted from the value of the albedo at time t-1.

- Variable 4 (var4): The value of the albedo of the composited pixel at time t.
- Variable 5 (var5): The value of the composited pixel in the middle-infrared channel (MIR) at time t subtracted from the value of the composited pixel in the middle-infrared channel (MIR) at time t-1.
- Variable 6 (var6): The value of the composited pixel in the middle-infrared channel (MIR) at time t.

Fisher's Linear Discriminant analysis method was used to detect burnt areas by multi-temporal change detection methods. S-PLUS commercial software was used to derive those variables that provided the maximum separability between burnt and not burnt land surfaces. Since the requirement of the GBA2000 project is solely burnt and not burnt surfaces, only the first discriminant axis is used to calculate the first discriminant variable (Johnson and Wichern, 1988). This variable is obtained by multiplying the coefficients of this axis with the values of the original variables described previously. The value of the first discriminant variable (FDV) is calculated by the following formula:

FDV = 0.0191017575562 * var1 – 0.0158523190767 * var2 – 0.0379260145128 * var3 + 0.0221815668046 * var4 + 0.0213510561734 * var5 – 0.0087660392746 * var6

Based on an analysis of the results, the FDV must be segmented between –0.5 and –1.0 as shown in Figure 16 to yield those pixels that are considered to be burnt areas. An analysis of the most suitable threshold of the FDV yielded a value of –0.8 (i.e. a FDV value greater then –0.8 would be flagged as being burnt in the final output product). However, for processing the large windows a value of –0.5 was chosen (to reduce commission errors).

To produce the burnt area maps, firstly pre-processed monthly composites were constructed. The composites are located in directories (inside the working directory) that indicate to the user the time period that the data has been composited (e.g. 200003_01_31). It is essential that data for both the month to be processed and the previous month be available. A c-shell script called gba_utl_processor.csh has been written to automate the production of burnt area products. A full description is given in Annex F. Typing this file name at the command line will yield the following information:

```
#########################################################
GBA2000 C shell: gba_utl_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_utl_processor.csh
<acronym>
<algorithm flag (1 = South Africa; 2 = Europe; 3 = Asia>
```

where the input parameters to this c-shell script are the acronym of the geographical region being processed and the code for the algorithm to be applied to the region of interest.

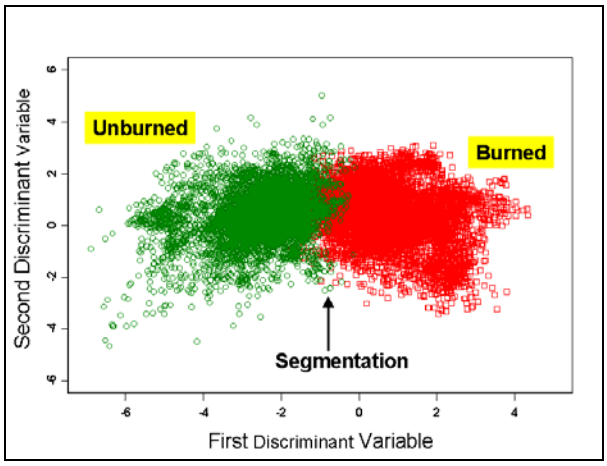


Figure 16: A plot showing the distribution of training pixels used in Fisher's Linear Discriminant analysis method. The value of the FDV is chosen to provide the best results between errors of commission and omission.

A text file is required to define the monthly composites to be processed. The text file utl_ba_months_list.txt is available and is located in the working directory. An example of this text file is:

```
# This text file is associated with the UTL BA algorithm
# Edit the months (m) that you want to process
# You do not need to consider the preceding month in the command line
# The interpretation and loading of the preceding month is automatic

# Full example:
# months = 20000101_31 20000201_31 20000301_31 20000401_31 20000501_31 20000601_31
20000701_31 20000801_31 20000901_31 20001001_31 20001101_31 20001201_31

# Actual files:
months = 20000701_31 20000801_31 20000901_31 20001001_31
```

The line that does not begin with a # symbol (and shown above in bold type) is the one that is interpreted by the controlling c-shell script. The c-shell script implements the burnt area algorithm on all of those monthly composites specified in the text file. By knowing the month that is to be processed, the program then knows which directory to look into to retrieve the

data from the preceding month. Once the names of the directories for the time t and t-1 are known then the program calls a second c-shell script that actually implements the burnt area algorithm. During the analysis of the performance of the algorithm, it was noted that commission errors were evident along coastal shorelines and those pixels located at the edges of water bodies. In addition, not all of these pixels were masked out in the pre-processing stages by the land cover map. A post-processing procedure was applied to the burnt area products that consisted of a two pixel dilated water (and non-burnable land surface) mask. This product is created in the working directory when the program is initially run. To produce the dilated water mask the UMD land cover mask and a reference image are used with the generic c program erode_mask.c (Annex A). The reference image is created using the generic c program make_raster.c (Annex A) and is defined as an array with the same image dimensions as the window of interest, and with all pixel values assigned a value of one. The dilation program erode_mask.c is applied twice to the land cover product to produce a two-pixel dilation.

The c-shell script that is run for each of the months being processed is called gba_utl_ba_algorithm.csh. As well as preparing the necessary data for processing, the script automatically calls an IDL program that contains the burnt area algorithm code. Typing this command at the prompt will yield the following information:

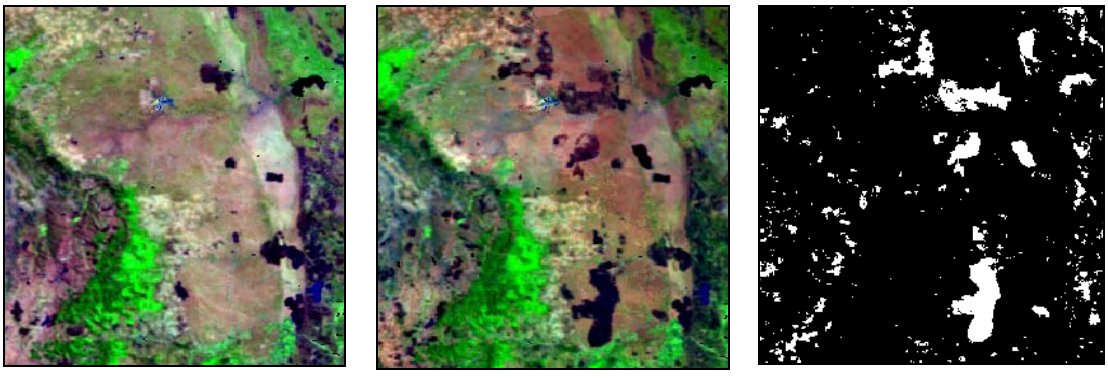
```
#########################################################
GBA2000 C shell: gba_utl_ba_algorithm.csh
Copyright JRC, 2001
Contact  person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_utl_ba_algorithm.csh
<acronym>
<file name at month m (e.g. 20000701_31)>
<file name at month m-1 (e.g. 20000601_31)>
<algorithm flag>
<path to start_compo.pro (IDL composite program)>
<path to IDL executable binary>
```

where, the full paths to the IDL binary executable and the IDL source code for the burnt area algorithm are located. A full description is given in Annex F.

The output files are written to a directory containing the two letter acronym (e.g. SA_utl_ba). To facilitate easy interpretation and reading of the input files by the IDL program, the necessary input files are joined together to form one BSQ file using the standard UNIX command cat. The input image files needed by the burnt area program are files B2, B3 and

MIR for the composite at time t (month m) and files B2, B3 and MIR for the composite at time t-1 (month m-1). The c-shell script then calls and runs an IDL program called gba_utl_af_ba_algorithm.pro (see Annex F). In this program, the values of the coefficients of the first discriminant variable and the value of the threshold of the FDV are set. In addition to writing the binary (0/1) image indicating those pixels that have been burnt during the month being analysed to disk, an array containing values of the first discriminant variable and a status map is produced. The value of the corresponding pixel in the status map indicates to the user those pixels for which there are composite data available at times t and t-1 (status file pixel value = 1), time t but not time t-1 or vice-versa (status file pixel value = 2), or for those pixels with no comparable data for both times (status file pixel value = 0). This indicates to the user that for a particular reason (most likely due to persistent cloud cover) data is not available for one of the monthly composites. In these cases, observations of the type of land cover these pixels represent is required to determine if additional processing is needed, for example, if composites from two months previous are used.

The burnt pixel image indicates for each pixel a value to indicate if the pixel is considered burnt. A value of one in the output image indicates an area is burnt, otherwise a zero is assigned to the value of the pixel. Figure 17 shows an example extracted from an area on the border between South Africa and Mozambique. The monthly composites for August and September 2000 are shown, displayed as an RGB image (Red = MIR composite band, Green = B3 composite band, Blue = B2 composite band). The resultant burnt area map is shown in the third figure using a threshold value of –0.8 on the first discriminant variable.



(a) (b) (c)

Figure 17: The Image composites from a small region in South Africa and Mozambique are shown for the months of August (a) and September (b) 2000. The burnt pixel map is shown on the right (c), where those areas shown in white are classified as being burnt during the month of September 2000.

After applying a threshold value to the first discriminant variable to derive an estimate of those pixels describing burnt areas over the large window of southern Africa, it was noted that some commission errors were observed. To remove these errors, tests on the spectral characteristics of burnt pixels were made after the threshold was applied. The following statements describe the cause of the commission errors and the method used to remove them:

- Problem: Remove those pixels that are saturated in the middle-infrared (MIR) channel at time t-1 (but not masked out in the MIR saturation mask) while preserving bright, vegetated surfaces. Solution: Potential burnt pixels are removed if MIR_{t-1} is > 670.
- Problem: False detections caused by shallow flooding of land at time t and free of water at time t-1 and false detections due to phenological changes in forested regions in South Africa. Solution: Potential burnt pixels are kept if DN values (logical AND) are $MIR_{t-1} > 89$ $B3_{t-1} > 249$, $MIR_t > 89$, $B3_t > 89$, $B2_t < 250$.
- Problem: To remove saturated values in the middle-infrared (MIR) channel at time t-1 not previously masked. Solution: Potential burnt pixels are removed if $MIR_{t-1} > 500$ and $B2_{t-1} < 100$.
- Problem: To remove pixels affected by the return of the middle-infrared signal (MIR) after being saturated at time t. Solution: Potential burnt pixels are removed if $MIR_t < 450$ and $B2_{t-1} > 170$.
- Problem: To remove falsely detected pixels caused by widely fluctuating composite values over forested regions. Solution: A burnt pixel signal remains burnt if $B3_t < 280$.

After processing, a dilated water mask is applied to the output files. This is done using the generic c program apply_mask.c (Annex A). The dilated water mask is multiplied with the burnt pixel and status images. To increase the temporal resolution of the products from a statement of the month of burning to a statement about the estimated day of burning for each burnt pixel determined by the UTL algorithm, the data file indicating the day of selection of the composite value (based on a minNIR criteria) is used. Assuming that a pixel is classified as being burnt then, because of the spectral characteristics of a burnt area, a pixel representing a burnt area is more than likely to be chosen for the composite. The day in the compositing period that the pixel is selected to form the composite is recorded in a file during the compositing procedures. This file can be used to estimate the day of burning for each

pixel classified as being burnt during the time period analysed. To enable direct comparison between these two pieces of information, the burnt pixel binary (0/1) is multiplied with the file indicating the day of burn. This results in all of those pixels that are burnt being preserved in the product of the two input files.

6.1.4 UTL post-processing procedure for Africa 1

The products resulting from the application of the UTL burnt area algorithm are needed to make the final determination of burnt areas. This final procedure is necessary because some pixels may be classified as being burnt in two consecutive months, which will lead to the inclusion of the pixel in the statistics for two months. Therefore, those pixels classified as being burnt must be removed from the results from analysis of following months. A commercial software tool, such as ENVI, can be used to do this task. Alternatively, tools are available that are described in Chapter 13 can be used. The monthly products are then merged together to produce a burnt area map for the year 2000.

6.2 UTL Europe module

An algorithm for burnt area detection in the woodlands and forests of Europe was developed by colleagues at UTL, Portugal (Silva *et al*. 2002b). The algorithm was developed using SPOT Vegetation imagery covering the Iberian Peninsula (Portugal and Spain) as shown in Figure 18. The algorithm was validated using Landsat TM imagery (the area covered by these images are also shown in Figure 18). The algorithm used is very similar to the algorithm described in detail in Section 6.1 and is not described again in detail in this section.

The pre-processing and mosaicking procedures are identical to the procedure described in Section 6.1.1 apart from differences in the cloud detection thresholds used. For Europe, the thresholds used for the detection of cloudy pixels are 230 for the B0 band and 250 for the MIR band. These thresholds were derived from in-depth analysis of the sensitivity of the degree of cloud removal obtained against the amount of bright non-cloudy pixels removed that could potentially burn. The compositing period is one month, based on a minimum near-infrared criteria (Section 4.1). The algorithm is activated using the same c-shell program that activates the algorithm for the Africa regional window. This program is called gba_utl_processor.csh and is described in Section 6.1 and Annex F. The user specifies the monthly composites to be processed by editing the text file utl_ba_months_list.txt. The

algorithm selection flag that is required at the command line of the c-shell script gba_utl_processor.csh should indicate that the UTL European algorithm is to be used.

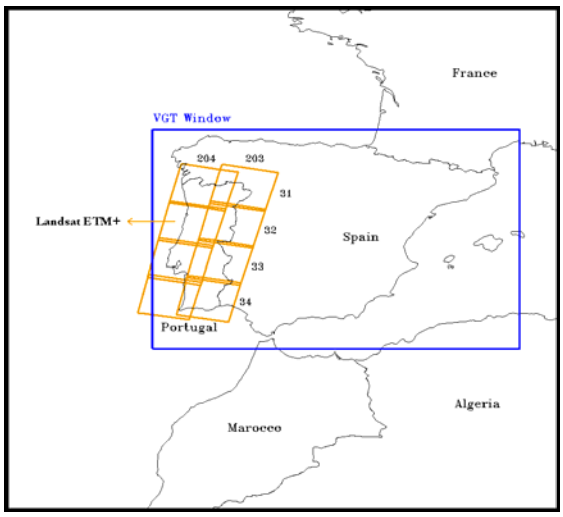


Figure 18: The Iberian Peninsula data window used by colleagues at UTL to develop the burnt area algorithm for Europe. The orange windows indicate the location of Landsat TM images that were used for validating the algorithm.

6.2.1 UTL burnt area algorithm procedure for Europe

For the detection of burnt areas with the UTL algorithm for Europe, six variables need to be calculated. Three are related to temporal differences (pre-fire conditions compared to post fire conditions) and three are related only to post fire conditions. They are described below, where t is the monthly composite being processed for burnt areas, and t-1 is the previous month's composite:

- Variable 1 (var1): The value of the composited pixel in the near-infrared channel (B3) at time t subtracted from the value of the composited pixel in the near-infrared channel (B3) at time t-1.
- Variable 2 (var2): The value of the composited pixel in the near-infrared channel (B3) at time t.
- Variable 3 (var3): The value of the NDVI of the composited pixel (defined here as (B3-B2)/(B3+B2) at time t subtracted from the value of the NDVI at time t-1.
- Variable 4 (var4): The value of the NDVI of the composited pixel at time t.
- Variable 5 (var5): The value of the SWVI of the composited pixel (defined here as (B3-MIR)/(B3+MIR) at time t subtracted from the value of the SWVI at time t-1.

- Variable 6 (var6): The value of the SWVI of the composited pixel at time t.

Fisher's Linear Discriminant analysis method was used to detect burnt areas by multi-temporal change detection methods. S-PLUS commercial software was used to derive those variables that provided the maximum separability between burnt and non-burnt land surfaces. Since, the requirement of the GBA2000 project is solely burnt and non-burnt surfaces, only the first discriminant axis is used to calculate the first discriminant variable (Johnson and Wichern, 1988). This variable is obtained by multiplying the coefficients of this axis with the values of the original variables described previously. The value of the first discriminant variable (FDV) is calculated by the following formula:

FDV = 0.00042644658242 * var1 – 0.00499185873196 * var2 + 4.73030662536621 * var3 + 2.84114623069762 * var4 + 3.57097363471985 * var5 – 3.44656109809876 * var6

Based on an analysis of the results, the threshold value of the FDV could not be defined clearly as a single value for the duration of the burning season in Europe. Rather different FDV values were needed for different months. An example is shown in Figure 19 shows the pixels used to derive the threshold value for the month of August (in the year 2000). An analysis of the most suitable threshold of the FDV yielded values of 2.1 for July (i.e. a FDV value greater then 2.1 would be flagged as being burnt in the final output product), 1.2 for August and 1.5 for September. After initial processing of the southern European window it was observed that these threshold values were not suitable for other Mediterranean regions. After intensive analysis of the results obtained using different thresholds the following values were used to derive the best results for the whole of the southern European region of interest, 2.1 for June and July, 1.75 for August and 2.0 for September. No analysis of burnt areas was done for any other month in the year 2000. The main reasons for this are that only a small amount of vegetation burning occurs outside of the northern hemisphere summer period and that large commission errors were observed over both agricultural (caused by ploughing/harvesting) and forested (caused by phenology) surfaces. Commission errors were observed in southern England (caused by agricultural practices over land surfaces wrongly indicated in the UMD product as being woodland etc.) during July and in the Alps (caused by phenology of trees) in September. In viewing the daily data it was seen that pixels in these areas flagged as being burnt were obviously not burnt, the change being due to the factors described above. Therefore, these regions (southern England in July and the Alps in September) were masked out from the results.

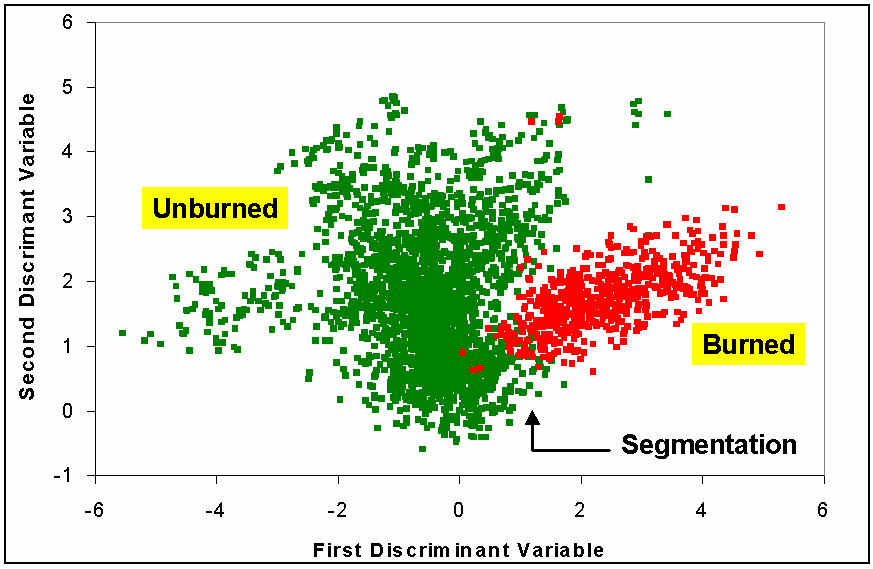
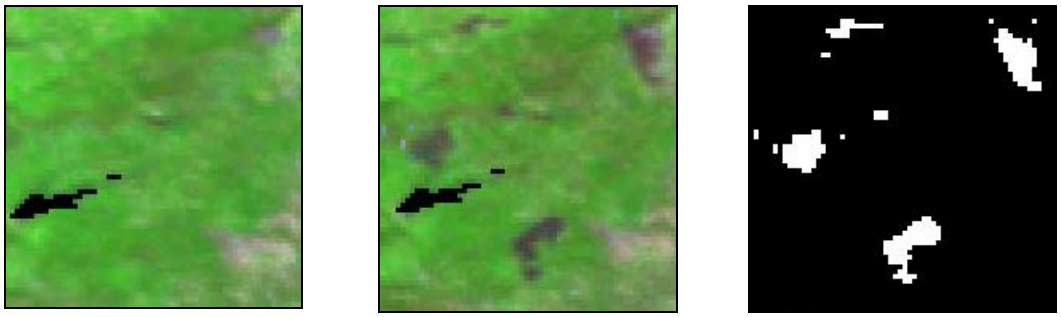


Figure 19: A plot showing the distribution of training pixels used in Fisher's Linear Discriminant Analysis to derive the maximum separability between those pixels that have burnt and those that have not burnt (for the month of August 2000). The value of the first discriminant threshold is chosen to provide the best results between errors of commission and omission over the algorithm test site (Iberian Peninsula).

The c-shell script that is called for each of the months being processed is called gba_utl_ba_algorithm.csh. As well as preparing the necessary data for processing, the script automatically calls an IDL program that contains the burnt area algorithm code. The output files are written to a directory containing the two-letter acronym for future reference (e.g. EU_utl_ba). The c-shell script then calls and runs an IDL program called gba_utl_eu_ba_algorithm.pro. In this program, the values of the coefficients of the first discriminant variable and also the value of the threshold of the FDV are set (for each month in the case of the latter parameter for Europe). The code for this IDL program can be found in Annex F. The value of the corresponding pixel in the status map indicates to the user those pixels for which there are composite data available at times t and t-1 (status file pixel value = 1), time t but not time t-1 or vice-versa (status file pixel value = 2), or for those pixels with no comparable data for either time (status file pixel value = 0).

The burnt pixel image indicates for each pixel a value to indicate if the pixel is considered burnt. A value of one in the output image indicates an area is burnt, otherwise a zero is assigned to the value of the pixel. Figure 20 shows an example extracted from a region in Portugal. The monthly composites for July and August 2000 are shown, displayed as an RGB image (Red = MIR composite band, Green = B3 composite band, Blue = B2 composite band). The resultant burnt area map is shown in the third figure.



(a) (b) (c)

Figure 20: An example of the burnt area product determined by the UTL algorithm. Image composites from a small region in Portugal are shown for the months of July (a) and August (b) 2000. After implementation of the linear discriminant analysis algorithm, using a FDV threshold of 1.2, the burnt pixel map shown on the right is produced (c), where those areas shown in white are classified as being burnt sometime during the month of August 2000.

After processing, a dilated water mask is applied to the output files. This is done using the generic c program apply_mask.c (Annex A). For the European algorithm the water mask is dilated only once, as opposed to twice for the African algorithm. The dilated water mask is multiplied with the burnt pixel and status images. To increase the temporal resolution of the products from a statement about the month of burning to a statement about the estimated day of burning for each burnt pixel determined by the UTL algorithm, the data file indicating the day of selection of the composite value (based on a minNIR criteria) is used. Assuming a pixel is classified as being burnt then because of the spectral characteristics of a burnt area, then a pixel representing a burnt area is more than likely to be chosen for the composite. The day in the compositing period that the pixel is selected to form the composite is recorded in a file during the compositing procedures. This file can be used to estimate the day of burning for each pixel classified as being burnt during the time period analysed. To enable direct comparison between these two pieces of information, the burnt pixel binary (0/1) is multiplied with the file indicating the day of burn. This results in all of those pixels that are burnt being preserved in the product of the two input files.

The post-processing procedures of the UTL algorithm for Europe are identical to those described in Section 6.1 and are not described here.

6.3 UTL Asia module

An algorithm for burnt area detection in the complex vegetated regions of northern Asia, outside of the main boreal forest zone was developed, based on a modified version of UTL algorithm for Africa. The algorithm needed several modifications due to the problems caused by phenology and snow cover. The pre-processing and compositing conditions remain the same as for the UTL Africa and Europe algorithms. Selection is made to apply the Asian algorithm at the command line of the c-shell script gba_utl_processor.csh (see Annex F). The program then calls an IDL program gba_utl_as_ba_algorithm.pro to apply the algorithm and generate the burnt area maps for each month specified in the text file utl_ba_months_list.txt.

The following conditions are applied to the burnt area algorithm for Asia based on the UTL algorithm for Africa.

- The threshold value of the FDV is set at –0.9, which is less strict than the threshold value imposed on the African datasets. This value was chosen after a qualitative sensitivity analysis exercise was undertaken to derive a satisfactory balance between omission and commission burnt pixels.
- The variable coefficients remained the same as for the UTL algorithm for Africa.
- Potential burnt pixels are removed if MIR_{t-1} is > 670. This removes the problem of saturated pixels in the MIR channel at time t-1.
- Potential burnt pixels are kept if DN values (logical AND) are $MIR_{t-1} > 89$ $B3_{t-1} > 249$, $MIR_t > 199$, $B3_t > 89$, $B2_t < 250$. This removes the problem of false detections caused by shallow flooding and phenology.
- Potential burnt pixels are removed if $MIR_{t-1} > 500$ and $B2_{t-1} < 100$. The removes the problem of saturated pixels not previously masked.
- Potential burnt pixels are removed if $MIR_t < 450$ and $B2_{t-1} > 220$. This removes the problem of returning saturated pixels in the MIR band at time t.
- A burnt pixel signal remains burnt if $B3_t < 320$ and $B3_{t-1} - B3_t > 90$. This removes the problems caused by widely fluctuating composite values over forested regions.

These spectral conditions are applied to all of those pixels satisfying the rules of the discriminant analysis method.

6.4 UTL Africa 2 module

A further algorithm for burnt area detection in the savanna grassland and woodlands of southern Africa was developed by colleagues at UTL, Portugal. The algorithm was developed and applied to windows CA, TA and SA as shown in Figure 3. The algorithm consists of three parts:

- Compositing of daily SPOT Vegetation images using a minimum near-infrared criterion over a monthly time period.
- Applying the burnt area detection algorithm based on Classification Trees and Regression Theory (CART).

The pre-processing and compositing criteria for this algorithm is identical to that described in Sections 6.1.1 and 6.1.2 respectively.

6.4.1 UTL burnt area algorithm procedure for Africa 2

A training data set was extracted from all the monthly composite images with significant burning activity and pooled together to create a single classification applicable to the entire fire season. A total of 30,075 pixels (14,924 of which corresponded to burned areas and 15,151 pixels to unburned areas) were extracted from the three processing windows for Africa (CA, TA and SA in Figure 3). Training data were chosen from all major vegetation types by visual inspection of pre-fire and post-fire color composite images, assisted by ancillary ATSR-2 active fire data (Arino *et al*., 2001) and the UMD global land cover product. Two composites were used to derive the burnt areas, t is the monthly composite being processed for burnt areas and t-1 is the previous month's composite. The threshold values are applied to the red (B2) and near-infrared (B3) channels at time t, the normalized difference water index (NDWI) (Gao, 1996) at time t and also the normalized difference vegetation index (NDVI) at time t. The value of the NDWI is computed by taking the ratio of the B3 value minus the MIR value over the B3 value plus the MIR value. The value of the NDVI is computed by taking the ratio of the B3 value minus the B2 value over the B3 value plus the B2 value. The temporal differences (t-1 minus t) of the NIR and NDWI values between successive composite images are also compared.

The burnt area mapping algorithm was developed using a supervised classification tree (Breiman *et al*., 1984). Classification trees are a non-parametric method based on binary

recursive partitioning. In our study, classification trees were applied to the SPOT Vegetation monthly composites in a change detection approach. The classification rules to determine a burnt surface are summarised in Table 1.

Table 1: The classification rules for the burnt surface class. A pixel is classified as burnt if it satisfies all the conditions of any classification rule. The variables used are the SPOT Vegetation Red (B2), NIR (B3) and middle-infrared (MIR) channels, and the NDVI and NDWI indices. $\Delta ijVARIABLE = VARIABLE_{time\ i} - VARIABLE_{time\ j}$, where times i and j refer to the composite images at time 1 and 2 respectively. The values shown are in DN.

Rule	Conditions (AND)
A	$nir_2 \leq 256.5$; $\Delta_{12}(nir) > 51.5$; $\Delta_{12}(nir) \leq 70.5$; $ndvi_2 \leq 0.309$
B	$nir_2 \leq 256.5$; $\Delta_{12}(nir) > 51.5$; $\Delta_{12}(nir) > 70.5$
C	$nir_2 > 256.5$; $ndvi_2 \leq 0.198$; $\Delta_{12}(ndwi) > 0.06$; $ndwi_2 \leq -0.19$
D	$nir_2 > 256.5$; $ndvi_2 > 0.198$; $nir_2 \leq 272.5$; $\Delta_{12}(nir) > 80.5$; $ndvi_2 \leq 0.290$
E	$nir_2 > 256.5$; $ndvi_2 > 0.198$; $nir_2 > 272.5$; $ndvi_2 \leq 0.231$; $\Delta_{12}(ndwi) \leq 0.076$; $\Delta_{12}(nir) > 69.5$; $red_2 \leq 213.5$;
F	$nir_2 > 256.5$; $ndvi_2 > 0.198$; $nir_2 > 272.5$; $ndvi_2 \leq 0.231$; $\Delta_{12}(ndwi) > 0.076$; $ndwi_2 \leq -0.212$

The burnt area algorithm is implemented using the c-shell script gba_utl_cart_processor.csh. The input variables to the script are as follows:

```
#########################################################
GBA2000 C shell: gba_utl_cart_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_utl_cart_processor.csh
<acronym>
<path to gba_utl_cart_ba_algorithm.pro>
<path to IDL executable binary>
<number of files to classify>
```

where, acronym indicates the region being processed (e.g. CA). The number of files contained within a text file indicating the files to be processed (monthly composites) is also given at the command line. A complete description is given in Annex F.

A text file indicating the files that are to processed should be copied or created in the working directory and given the name gba_utl_month_list.txt and resembles the following:

```
CA_19991201_31_b2_all_minNIR
CA_19991201_31_b3_all_minNIR
CA_19991201_31_mir_all_minNIR
CA_20000101_31_b2_all_minNIR
CA_20000101_31_b3_all_minNIR
CA_20000101_31_mir_all_minNIR
CA_2000010131_all_minNIR
…
CA_20001201_31_mir_all_minNIR
CA_2000120131_all_minNIR
```

The format of the text file (in the example shown) informs the program that the composite at time t-1 comprises of data from he month of December 1999 and the composite at time t comprises of data from the month of January 2000. The seventh file in the list indicates the name of the output file that will be written to a new directory named gvm_ba. In the case of sub-Saharan Africa for the year 2000, the values of the number of files to be processed are 56, 63 and 84 for the windows CA, SA and TA respectively. Into the directory utl_ba monthly burnt area maps, with coding that shows a pixel value of one if the pixel burnt and zero otherwise, are written. The c-shell script calls an IDL program, containing the burnt area algorithm. This IDL program is called gba_utl_cart_ba_algorithm.pro and is described in Annex F.

6.4.2 UTL post-processing procedure for Africa 2

The first post-processing procedure is the multiplication of each monthly burnt area product with a one-pixel expanded water mask. This removes false detections that surround large water bodies. In addition, the masking of the Arabic Peninsula that is present within window CA was necessary. Pixels detected as being burnt in this region of Saudi Arabia and the Yemen were found to be erroneous. The tools used to generate and apply these masks are described in Annex A. Unique monthly products were then derived using tools described in Chapter 13. For window CA, this procedure was only undertaken until the month of September 2000. In October, the burning season commenced and it was assumed that the same pixel detected as being burnt in January or February 2000 could also burn again at the beginning of the next dry season. The monthly products for the windows CA, SA and TA were then merged together to produce the burnt area map for sub-Saharan Africa for the year 2000.

7 NRI algorithm module for south-west Africa

The Natural Resource Institute (NRI), the GBA2000 partner comprised of Luigi Boschetti and Stephane Flasse, located at the University of Greenwich in the UK, were responsible for producing an algorithm for the detection of burnt areas from SPOT Vegetation data. The algorithm was developed, calibrated and validated over Botswana and Namibia (south-west Africa). In the final GBA2000 product the algorithms was not used. After an in-depth analysis and comparison of the results of the NRI and UTL algorithms for southern hemisphere Africa it was decided based on an assessment of the errors of omission and commission that the UTL algorithm would be applied in this region. However, details of the algorithm are described here.

7.1 NRI Africa module

An algorithm for burnt area detection in the savanna grassland and woodlands of south-west Africa was developed by Boschetti *et al*. (2002). The study area can be seen in Figure 21. The image shows a SPOT Vegetation S1 product (Red = middle infrared (MIR) band, Green = near-infrared (B3) band, Blue = red (B2) band). Also shown are the regions where Landsat TM imagery was used to validate the burnt area algorithm. The fire season in this region of southern Africa normally starts in May or June and ends in December or January.

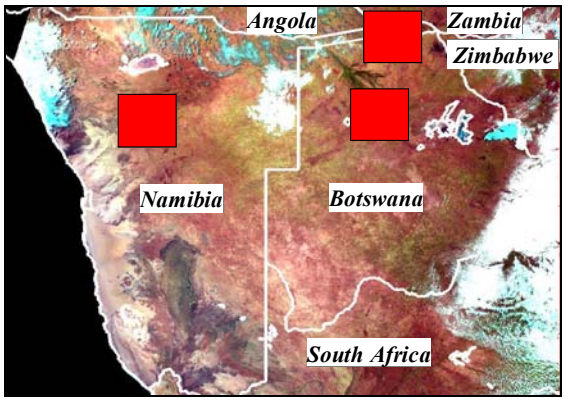


Figure 21: The location of the study area in south-west Africa used to develop the burnt area algorithm by the NRI partner. The red squares show the location of the Landsat TM imagery.

The burnt area algorithm exploits the fact that a vegetation fire drastically changes the spectral characteristics of the local environment and by looking at pre-burn and post-burn

conditions these changes, that are greater than changes brought on by other factors, can be identified. The main features and assumptions of the approach taken are:

- To apply a change detection algorithm it is assumed that the changes in the spectral radiance due to the land cover change induced by a fire are significantly greater than the changes due to other factors (Ingram *et al*., 1981). The spectral channels corresponding to the near-infrared (B3) and middle-infrared (MIR) are used to detect the changes resulting from a fire event, following the results and recommendations of Trigg and Flasse (2000).
- The change detection technique is based on the comparison between a new (or current) image and a reference image. The latter image is initiated at the start of the processing (i.e. at the beginning of the fire season) and updated at every step (each time a new image is available) by those pixels that are defined as being clean (i.e. those pixels not affected by clouds, shadows or other perturbing factors).
- The algorithm takes into account the variation in spectral signals caused by changes in the viewing geometry (observation angle) of the sensor. Analysis by the NRI group showed that the amplitude of this variation in spectral signature is in the same range of variation as the difference between adjacent pixels (one burnt and one not burnt) after a fire event (i.e. up to five per cent of the overall reflectance value). The variation is explained by looking at the viewing geometry cycle of the SPOT Vegetation sensor. The viewing conditions of every geographic point vary over a 26-day cycle but every five days the viewing conditions are very similar. It was decided to calculate the temporal change indices with a sampling period of five days to minimise the variations due to viewing conditions. Further details can be found in Boschetti *et al*. (2002). Operationally, this means that the whole dataset is split into five separate sets of images with similar viewing conditions that are then analysed independently. The results of the processing of each series are integrated and further analysed to derive a final burnt area product.
- For the algorithm to work, it is important to remove pixels contaminated by cloud, smoke and cloud shadow from the dataset. The contaminated data are masked out using a two-stage process. The first stage is to apply the pre-processing module developed specifically for the GBA2000 project (see Chapter 3). The second stage is to make further tests for thin cloud, fire smoke and sudden increases in reflectance

using temporal (and not spectral) based tests. These programs are discussed in the following sections.

7.1.1 NRI pre-processing procedure

The pre-processing requirements of the NRI burnt area algorithm for southern Africa are similar to that of the UTL algorithm developed in the study area of Mozambique. The aim of applying the pre-processing module to the dataset is to produce a clean image product that is not contaminated with pixels that might lead to errors of commission in the final image product. The c-shell script gba_preprocessor.csh (Chapter 3) was used to undertake the pre-processing of the southern Africa dataset. The pre-processing module was used to remove those pixels acquired at extreme viewing zenith angles, those affected by saturation in the MIR channel, those contaminated by cloud pixels (using threshold values of 230 in the blue (B0) channel and 250 in the middle-infrared (MIR) channel), those surrounding the pixels defined as being cloud, those that are likely to be affected by cloud shadow and those characteristic of non-vegetated land surfaces. An example of the input command to the c-shell script gba_preprocessor.csh is:

gba_preprocessor.csh SA 1 60 1 1000 1 4 1 230 250 1 1 10000 75 1 all 1

The pre-processing module is applied to all of the dates specified in the text file input_dates.txt. Complete descriptions of the GBA2000 pre-processing procedures are given in Chapter 3.

7.1.2 NRI burnt area algorithm procedure for Africa

The first stage in the burnt area algorithm procedure is to present the input data (pre-processed, daily S1 products) in a format that can be easily read in and processed by the burnt area algorithm. The second stage is to apply the algorithm and generate the results. In a series of images for consecutive days the program must identify those datasets that were acquired five days apart. For each series, the algorithm is applied independently. For each image in the series, the pre-processed data are analysed using the following method:

- For the first image in a series, all clean pixels are assigned to the reference image.
- For the next image in a series, the difference between the new image and the reference image is calculated for specified channels on a pixel-by-pixel basis.
- If the difference between the values is positive (i.e. an increase in reflectance), when subtracting the pixel value of the reference image from the pixel value of the image

under analysis and this difference is greater than a fixed threshold, then the pixel is flagged as being contaminated and removed from the analysis at this stage (i.e. it cannot be placed within the reference image).

- If the differences between the two images in the near-infrared (B3) and middle-infrared (MIR) bands are negative (i.e. a decrease in reflectance) and these differences are greater than fixed thresholds, the pixel is flagged as being potentially burnt. The date that the pixel is detected as being burnt is also recorded. This pixel is then removed from further consideration when processing images acquired at later times within the sampling period.
- The reference image is then updated. The value that the pixel is assigned is that of the image under analysis, except for those pixels identified as being cloud, cloud shadow or burnt in the previous stages outlined above. The reference image, therefore, represents at any time the latest clean observation of a pixel that has not previously been detected as being burnt.

The criteria used to determine those pixels that are burnt or contaminated, in addition to those masked during the pre-processing stage, in the daily images are as follows (all pixel values refer to the original S1 pixel DN encoding in the range 0 to 2000):

- Pixels are flagged as being cloudy if the following conditions are all satisfied (a logical AND). The value of the pixel in the blue (B0) channel is greater than 180 AND the value in the red (B2) channel is greater than 220 AND the value in the near-infrared channel is greater than 320 AND the value in the middle-infrared (MIR) channel is greater than 500.
- Pixels are flagged as erroneous, due to increases in reflectance, if the following conditions are satisfied. The difference between the pixel values in the blue (B0) channel (reference value subtracted from the current image value) is greater than 60 OR difference between the pixel values in the red (B2) channel is greater than 60 OR if both values of the blue (B0) and middle-infrared (MIR) channels of the image being analysed are both equal to 0, when the value of the middle-infrared (MIR) channel in the reference image is non-zero (a logical AND).
- Pixels are identified as being potentially burnt if the following conditions are satisfied (after all tests for contaminated pixels have been applied). The difference between the pixel values in the middle-infrared (MIR) channel (current image value subtracted

from the reference image value) is greater than 100 AND the difference between the pixel values in the near-infrared (B3) channel is greater than 100 AND the current image being processed contains non-zero values.

The burnt area algorithm has been coded using the IDL programming language and is controlled automatically by a c-shell script called gba_nri_processor.csh. A complete description of this program is given in Annex G. By typing this program name at the command line, the following information will appear:

```
##########################################################
GBA2000 C shell: gba_nri_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
##########################################################
Syntax: gba_nri_processor.csh
<acronym>
<path to start_compo.pro (IDL composite program)>
<path to IDL executable binary>
<specify the output filename (nri_march)>
<Reference image already created (0 = no = create reference, 1 = yes)>"
<Name of reference image to use (e.g. 20000101_31) if previous input = 1; otherwise = 0>"
```

where, the full paths to the IDL binary executable and the IDL source code for the burnt area algorithm are required. The name of the output files needs to be specified as well as an indication to whether the user wants to utilise with burnt area algorithm while creating and updating the reference image continuously or to utilise a reference image previously created. A full description is given in Annex G.

During the algorithm's development and testing phases, two methods of deriving the reference image were explored. It was discovered that, depending on the timing of the fire season and the availability of data, some burnt areas were being missed, due to the time it took to completely fill the reference image with clean data. If a reference image could be prepared beforehand with values indicative of pre-fire season conditions (for example using a compositing technique) then the performance of the algorithm would improve. This is especially so over regions with short fire seasons. Two variations of the algorithm were developed. One algorithm would create the reference image using the sequence of observations made over a five day time period (as specified in the original document by Boschetti *et al*. (2002)). The second algorithm relied on a reference image already being available before the first image in the series is analysed. The latter method has the advantage that comparisons can immediately be made between images from the first day of the fire

season. The user selects the algorithm that is to be used at the command line of the c-shell script gba_nri_processor.csh. The code for both of these algorithms are available in Annex G. A description of the original algorithm, that starts with no reference image and continuously updates it, is given in this section.

The c-shell script temporarily moves and renames all of the required spectral files into the working directory. A list, in the form of a text file, is created containing all of the dates available to be processed. The selection of the burnt area algorithm is now made. Depending on the availability of a reference image already being available or the user's need to create the reference image, a code flag is given at the command line of the controlling c-shell script. If no reference image is available, then the IDL program gba_nri_ba_algorithm_a.pro is activated automatically (see Annex G). If a reference image is available, created for example by compositing, then the IDL program gba_nri_ba_algorithm_b.pro is activated automatically (see Annex G).

The output products for each of the data series include a copy of the reference image made after the last day in the time series has been processed. All four channels of the reference image are written to file. An image containing those pixels determined as being potentially burnt is also written to file. The value of a potentially burnt pixel represents the location (numbered sequentially) within the time series when the pixel was first detected as being burnt. For example, the burnt area algorithm was initiated one month before the beginning of the official fire season, for example on the 1st of April. For the first five day sequence, the following code would be given to the dates that are processed, 1st April = 1, 6th April = 2, 11th April = 3, and so on. For the second 5-day sequence, the following code would be given, 2nd April = 1, 7th April = 2, and so on until the end of the fire season. This code is used to assign a value to the pixel when it is determined as being potentially burnt. Therefore, from the pixel values of the potentially burnt area image, it is possible to identify the exact day (and month) that the pixel was first identified as being burnt. The analysis and integration of each series of data is implemented in the post-processing module of this algorithm, presented in the next section. However, it is important to note here that the user needs to be aware of any missing data, which could lead to errors in correctly interpreting the pixel values in the potential burnt area image products.

7.1.3 NRI post-processing procedure for Africa

The requirements of the post-processing module of the NRI algorithm for Africa is to derive actual burnt area products from potential burnt area products. This is achieved through analysis of the five time series of potentially burnt area images. The following parameters can be extracted from the time series of data to assist in the determination of actual burnt areas:

- The total number of detections for each pixel over the five series of data (up to a maximum of five). This number gives us more confidence in that the pixel is burnt if it is detected more than once throughout the fire season.
- The date (day or month) of the first time that the pixel is detected as being potentially burnt. This value can be compared with existing data (from previous years) to visually confirm that the spatial and temporal distribution is likely to be correct.
- The variation in the date of detections if the pixel is detected more than once in the five series. For example, if in one five day series the pixel is detected as being burnt in the month of May, burnt in August in another series and in November in a third, then this does not fit very well with the theory of burnt area detection as the dates are too spaced out. On the other hand, if three detections were obtained just in the month of July for another pixel, then it is assumed that this is more likely to be a true burnt area.

It is to be expected that anomalies will occur. These anomalies may be due to persistent cloud cover in some areas reducing the number of observations made. Errors in the data may force pixels to become identified as burnt areas when not, resulting in that pixel to be removed from further consideration by the algorithm. A post-processing program was developed to give the user maximum flexibility when constructing the final burnt area map.

The post-processing module of the NRI algorithm is controlled by the c-shell script gba_nri_postprocessor.csh. The input variables to the script are as follows:

```
#########################################################
GBA2000 C shell: gba_nri_postprocessor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_nri_postprocessor.csh
<acronym>
<input file name (e.g. nri_jan_may)>
<apply a 1-pixel dilated water mask to the final burnt area map (0=no, 1=yes)>
<number of total detects for burnt areas (e.g. 2)>
```

In the example shown, only those pixels detected two times or more (>=) will be flagged as being burnt in the final burnt area product. The user has the opportunity to apply a one pixel dilated water (non-vegetated) mask to the results to remove any errors of commission that might exist. A complete description of this c-shell script is given in Annex G.

An example of the command used to control the post-processing module is:

gba_nri_postprocessor.csh SA nri_ba 1 0 1 2

The first step in the post-processing module is to recode all five series of results into the respective month that the pixel was detected as being burnt. This is achieved by editing a text file and saving it into a directory called recode_text_files, located inside the working directory. This text file is interpreted by the generic c program binary_remap.c (Annex A). For each of the potential burnt area products, the program reads in the results from the previous module and depending on the values in the text file, writes out a product with the pixel value corresponding to the month during the year 2000 that the pixel was first detected as being burnt. Using the example shown in the previous section, if the third series was being analysed (start date equals the 3rd of April 2000) and the pixel value is nine indicating that the pixels was identified as being burnt in the 9th image in the series, then the date of detection of a potentially burnt area is actually the 13th of May 2000. The generic c program binary_remap.c creates a new output file. The new pixel value will be 2, corresponding to May, the 2nd month being processed. Note that the value of the pixel cannot be represented as a 5 (i.e. May, the 5th month), because if the processing is made between different years then it restricts the use of comparison tests made later to determine if the detections have been made close together (as described in the third bullet point above). This situation will be corrected later in the post-processing. A different text file is required for each of the five series (series 0 to series 4). An example is now given of the file SA_cycle_1_remap.txt used to process the second series of data between the periods 1st January 2000 and 31st March 2000 is:

```
0 0
1 1
2 1
3 1
4 1
5 1
6 1
7 2
8 2
9 2
10 2
11 2
```

12 2
13 3
14 3
15 3
16 3
17 3
18 3

An example of the command used to run the c program to recode each series of data is:

binary_remap SA_nri_ba_0 SA_nri_ba_0_remap SA_cycle_0_remap.txt $pixels $lines

where, SA is an example of the acronym of the geographical region being processed. A complete description of this program is given in Annex A.

Once the new, recoded image is available for each series the next step is to integrate the series together to form the final burnt area map. The c program nri_ba_overlay.c is used to undertake this task. The program also makes a series of tests before producing the final burnt area map. One of the input parameters to this program is a value of the threshold to be applied to the total number of detections for each pixel within the five series of data. If the number of detections is equal to or greater than this value then the pixel is temporarily classified as being burnt. The second test looks at the variation in the timing of the burn for the five series within the time series. For a pixel to be classified as being burnt in the final output, all of the detections must be within one month either side of, what is defined as being the month that the pixel was first detected as being potentially burnt as the program integrates each series in turn. An example of the command used to integrate the series of data, to produce the final burnt area image, is:

nri_ba_overlay $pixels $lines SA_nri_ba_0_remap SA_nri_ba_1_remap SA_nri_ba_2_remap SA_nri_ba_3_remap SA_nri_ba_4_remap SA_nri_ba $no_detections

where, the number of detections tells the program to ignore all of the potentially burnt areas detected only once during the fire season to be ignored. A complete description of this program is given in Annex G.

The next step in the post-processing module is, if specified, to apply a dilated water mask to the final burnt area map. It was seen that some errors of commission are located at the edges of water bodies so, by applying a dilated water mask, these are removed. The final burnt area product contains information on the time of burning for each pixel identified as being burnt. The penultimate step is to extract from the geographical region the unique region of interest (without a twenty pixel window buffer). The unique region covered by the final burnt area

map is extracted to a new directory called $acronym_nri_ba_results, using the generic c program snip.c (see Annex A). The final step in the post-processing module is to construct individual burnt area images for each month of the year 2000, and also to create one image for the whole of the year 2000. Pixels within these images contain a value of one if the pixel is burnt and zero if the pixel is not burnt. The generic c program binary_remap.c is again used for this task. The final image product is recoded to produce monthly burnt area products and also a yearly burnt area product. For each month to be recoded, a text file is needed (located in the directory recode_text_files). These files need to be edited so that the value of the pixel in the integrated burnt area image corresponds to the individual month that is being extracted. Within the newly created directory $acronym_nri_results, are situated monthly and year 2000 burnt area maps for the geographical region under investigation. The same post-processing procedure can be applied to the potential burnt area products derived from both variations of the burnt area algorithm (i.e. with a reference image previously created or created by the burnt area algorithm, as described in the previous section).

8 GVM (Stoppiana) algorithm module

The GVM group, working in the Joint Research Centre and comprised of Daniela Stroppiana and Jean-Marie Gregoire, were responsible for producing an algorithm for the detection of burnt areas from SPOT Vegetation data over continental Australia and Tasmania (window AU in Figure 3). The algorithm is composed of a set of thresholds applied to each pixel's value in the Vegetation instrument's spectral channels, two spectral indices and their temporal difference. The threshold values have been derived by means of a supervised classification methodology based on the Classification And Regression Trees (CART) theory (Stroppiana and Gregoire, 2001; Stroppiana *et al*., accepted in IEEE TGRS).

Fires can occur at anytime of the year in Australia. Large scale controlled burning of vegetation takes place annually in the north of Australia between April and November (the northern dry season). Fires in the south of Australia normally occur during the southern hemisphere summer (i.e. from November to April). However, it is possible for vegetation to burn outside of these periods as well. To capture all of the vegetation burning in Australia it was necessary to process the whole of the year 2000 dataset. The burnt area algorithm consisted of four parts:

- Pre-processing of daily SPOT Vegetation products to yield clean data.
- Applying a minimum near-infrared (NIR) compositing criteria over a ten-day period.
- Applying the burnt area algorithm using thresholds derived from a supervised classification methodology based on the CART methods.
- Filtering and summation of the ten day burnt area maps into monthly products.

The extraction of the Australian datasets from the tape archive is undertaken using programs described in Chapter 2.

8.1 GVM (Stroppiana) pre-processing module

The pre-processing requirements of the GVM burnt area algorithm for Australia are the effective removal of cloud and cloud shadow, non-vegetated surfaces, saturated pixels in the MIR band and pixels acquired at extreme viewing zeniths from daily SPOT Vegetation datasets. The c-shell script gba_preprocessor.csh is used (see Chapter 3) because this script satisfied all of the above requirements. The thresholds used for the detection of pixels that are

cloudy are 180 for the B0 band and 180 for the MIR band. The cloud mask was also dilated by one pixel before calculation of the cloud shadow mask. An example of the command used to pre-process the data, for all of those dates indicated in the text file input_dates.txt, is:

gba_preprocessor.csh AU 1 60 1 1000 1 4 1 180 180 1 1 10000 75 1 all 1

For each day, the pre-processed data would contain data that were masked for cloud, cloud shadow (assuming a cloud height of 10 km), saturation in the MIR channel (all pixels with a DN > 1000), non-vegetated and water surfaces and pixel data acquired at extreme viewing zeniths (> 60 degrees). The threshold values for the cloud masking were extensively tested for the sensitivity to remove cloudy pixels but also to preserve those pixels that represented bright, non-cloudy pixels on the Earth's surface.

8.2 GVM (Stroppiana) compositing procedure

The pre-processed SPOT Vegetation data was composited over a time period of ten days according to a minimum near-infrared criteria (Section 4.1). The text file input_dates.txt was edited to reflect the time period to be composited. To make the composites, the c-shell script gba_min_nir_composite.csh was used. An example of the command used is:

gba_min_nir_composite.csh AU ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin all d 1

This command indicates that level two products (including, for each pixel, the number of cloud free days, the day in the time period from which the pixel was selected for the composite and the status image) were produced from fully pre-processed data over a decadal time period.

8.3 GVM (Stroppiana) burnt area algorithm procedure

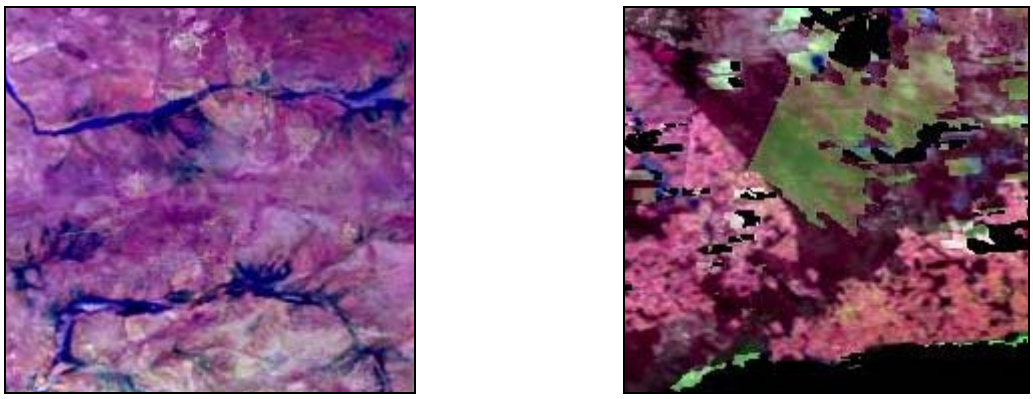
The burnt area algorithm has been developed by means of a supervised classification technique based on the Classification and Regression Trees (CART) theory (Breiman *et al*., 1984) already successfully applied for land cover and burnt area mapping using satellite images (Hansen *et al*., 2000; Pereira *et al*., 2000). A training set is used to derive a decision tree from which a set of classification rules (i.e. spectral thresholds) are extracted. The threshold values are applied to the near-infrared (B3) and middle-infrared (MIR) channels, an albedo-like index, referred to in this report as albedo, defined here as the arithmetic mean of the bands B2 and B3 reflectance (Saunders, 1990), and the normalized difference water index (NDWI) (Gao, 1996). The value of the NDWI is computed by taking the ratio of the B3 value minus the MIR value over the B3 value plus the MIR value. The temporal differences of

these values between successive composite images are also compared. The detection of a change over a time period in the spectral signal can reduce the likelihood of misclassification between burnt surfaces and other targets with a similar spectral signature (e.g. water, land and other mixed pixels). The algorithm presented here exploits each pixel's spectral value in three consecutive composite images (times 1, 2 and 3) to map pixels burnt between the first and the second composite. The third composite image is used to confirm the presence of a burnt area. The algorithm is applied only to those pixels that are clear in all three composites. The output burnt area map is a classification of the composite image at time 2 into burnt, not burnt and unknown classes (the unknown class being those pixels masked out in any of the composite). The classification rules to determine a burnt surface are summarised in Table 2.

Table 2: The classification rules for the burnt surface class. A pixel is classified as burnt if it satisfies all the conditions of any classification rule. The variables used are the SPOT Vegetation NIR (B3) and middle-infrared (MIR) channels, and the albedo and NDWI indices. $\Delta ijVARIABLE = VARIABLE_{time\ i} - VARIABLE_{time\ j}$*, where time i, j, k refer to the composite images at time 1, 2, and 3 respectively. The values shown are in reflectance.*

Rule	Conditions (AND)
A	$nir_2 \leq 0.130$; $\Delta_{21}(nir) \leq -0.045$; $ndwi_3 \leq -0.05$; $\Delta_{32}(nir) \leq -0.10$
B	$nir_2 \leq 0.130$; $\Delta_{21}(nir) \leq -0.045$; $ndwi_3 \leq -0.05$; $-0.10 < \Delta_{32}(nir) \leq 0.00$; $\Delta_{32}(mir) > -0.007$
C	$nir_2 \leq 0.130$; $\Delta_{21}(nir) \leq -0.045$; $ndwi_3 \leq -0.05$; $\Delta_{32}(nir) > 0.00$
D	$nir_2 \leq 0.130$; $\Delta_{21}(nir) \leq -0.045$; $ndwi_3 > -0.05$; $nir_3 \leq 0.129$
E	$nir_2 \leq 0.130$; $\Delta_{21}(nir) > -0.045$; $\Delta_{21}(mir) \leq -0.024$ $\Delta_{32}(albedo) \leq -0.006$; $\Delta_{32}(nir) > -0.011$
F	$nir_2 > 0.130$; $\Delta_{32}(mir) \leq -0.314$; $albedo_2 \leq 0.142$
G	$nir_2 > 0.130$; $\Delta_{32}(mir) > -0.314$; $nir_2 \leq 0.140$; $ndwi_3 \leq -0.094$ $\Delta_{21}(nir) \leq -0.047$

To determine if a pixel is classified as being burnt, any of the conditions stated in Table 2 can be satisfied. The conditions are mutually exclusive, in that a pixel cannot satisfy both rules A and C. The threshold values were derived from the supervised classification of images where vegetation burning had occurred in the northern part of Australia. Some commission errors are believed to exist in the final products. These are due to a number of reasons such as flooding of normally dry valleys, agricultural practices and when the quality of the composite image was contaminated in regions experiencing cloudy and wet conditions. Examples of these problem areas are shown in Figure 22. In the left image, flooded valleys in the centre of the image are detected as burnt areas due to their changing spectral conditions. In the right image the composite image is not very homogeneous leading to false detection in the green region at the centre of the image in the next composite image.



(a) (b)

Figure 22: Examples of sources of commission error in the GVM algorithm for Australia. (a) shows flooded valleys with change characteristics similar to that of burnt areas. (b) shows compositing noise caused by unstable climatic conditions at the time of compositing.

A search of the Internet for information concerning the timing of the wet season in different parts of Australia indicated that the timing and magnitude of precipitation events were not consistent from year to year. Also obtaining good quality maps of this information was nearly impossible. The algorithm was developed mainly over land cover types that could be grouped together under a tropical savanna umbrella. The tropical savanna regions of Australia are well mapped and even have a dedicated web site providing information about a range of subjects concerning these regions (http://savanna.ntu.edu.au). A mask was made using information from this website (of the geographical distribution of tropical savannas). A coarse scale map was used to derive a co-registered (using coastlines as tie-points) product in the same geometry as the burnt area products. Resampled versions of these masks are shown in Figure

23, the tropical savanna mask is shown on the left and the non-tropical savanna mask is shown on the right. For the months of February and March 2000, situated in the middle of the wet season in the tropical region, only commission error was observed (due to flooding) and so these months were removed completely from the final product using the tropical savanna mask.

(a) (b)

Figure 23: An illustration of the land cover masks used to distinguish between tropical savannas (shown in white in (a)) and non-tropical savannas (shown in white in (b)). The masks have been produced using geographical data derived from the Internet.

The burnt area algorithm is implemented using the c-shell script gba_gvm_processor.csh. The input variables to the script are as follows:

```
#########################################################
GBA2000 C shell: gba_gvm_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#########################################################
Syntax: gba_gvm_processor.csh
<acronym>
<path to gba_gvm_algorithm.pro>
<path to IDL executable binary>
<number of files to classify>
```

where, acronym indicates the region being processed (e.g. AU). The number of files contained within a text file indicating the files to be processed (ten day composites) is also given at the command line. A complete description is given in Annex H.

A text file indicating the files that are to processed should be copied or created in the working directory and given the name gba_gvm_tenday_lists.txt, and resembles the following:

```
AU_19991221_31_b2_all_minNIR
AU_19991221_31_b3_all_minNIR
AU_19991221_31_mir_all_minNIR
AU_20000101_10_b2_all_minNIR
AU_20000101_10_b3_all_minNIR
AU_20000101_10_mir_all_minNIR
AU_20000111_20_b2_all_minNIR
AU_20000111_20_b3_all_minNIR
AU_20000111_20_mir_all_minNIR
AU_2000010110_all_minNIR
…
AU_20001221_31_mir_all_minNIR
AU_2000121121_all_minNIR
```

The format of the text file (in the example shown) informs the program that the composite at time t-1 comprises the last ten days of 1999, the composite at time t comprises the first ten days of January 2000 and the composite at time t+1 comprises the middle ten days of January 2000. The tenth file in the list indicates the name of the output file that will be written to a new directory named gvm_ba. In the case of Australia for the year 2000, the value of the number of files to be processed that is entered into the above c-shell script is 350 and not 360 (because the burnt area maps for the final ten days of the year 2000 are not available). Into the directory gvm_ba are written ten day burnt area maps, with coding that shows a pixel value of one if the pixel burnt and zero otherwise. The c-shell script calls an IDL program, containing the burnt area algorithm. This IDL program is called gba_gvm_ba_algorithm.pro and is described in Annex H.

8.4 GVM (Stroppiana) algorithm post-processing procedure

The ten-day burnt area products are firstly filtered using a 3x3 median filter and then composited to create monthly burnt area products. After filtering, a pixel in the monthly product is determined as being burnt if is has been detected as being burnt in only one of the three ten day products making up the monthly composite. Once all of the monthly maps have been produced a final check for pixels that have been detected as burnt more than once in the year 2000 are made. If any of these double detection pixels exist they are removed from the results. Because of the fire seasonality in Australia it was considered unlikely that the same areas would have burnt twice during the year 2000. The creation of monthly products from ten-day results is achieved using the c-shell script gba_gvm_postprocessor.csh (see Annex H) that requires the following input parameters:

```
#######################################################
GBA2000 C shell: gba_gvm_postprocessor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_gvm_postprocessor.csh
<acronym>
<path to gba_gvm_make_monthly_maps.pro>
<path to IDL executable binary>
```

The program calls an IDL programs that filters the burnt are pixels using a 3x3 median filter and then composites the ten-day composites according to the criteria described above. Note that in the absence of data for the first ten days of the year 2001, the algorithm processes only two, ten-day composites for the month of December 2000. The IDL programs that is used for this processing is called gba_gvm_make_monthly_maps.pro and is described in Annex H.

The final step is to mask with the tropical savanna land cover product the burnt area maps for the months of February and March 2000. After some feedback was given by our Australian partners on the accuracy of the burnt area products and, in particular, not capturing completely some burnt areas we decided to apply a burnt area expansion program. The generic c program ba_expand.c (Annex A), works on a pixel by pixel basis. If a pixel is indicated as being burnt and also six of the eight pixels in a surrounding 3x3 window are also indicated as being burnt then all eight pixels are flagged as burnt. This essentially fills any gaps in the data while leaving small clumps of burnt areas unaffected. A one pixel expanded water mask is also applied to the monthly composites. The programs used to derive these unique monthly burnt area products are called unique_ba_maps_1.c and unique_ba_maps_2.c and are described in Chapter 13.

9 CCRS algorithm module

The CCRS group, comprising Rob Fraser (and colleagues), was responsible for production of a burnt area algorithm for forested regions of Canada. After extensive testing it was proposed that the algorithm was applied to forested regions of the USA (C1, C2 and US in Figure 3). The underpinning of the procedure is multi-temporal change detection based on a multiple logistic regression model and satellite-based change metrics (Fraser *et al*., submitted). Multiple logistic regression and artificial neural network approaches were considered for this task owing to their ability to model the probability of a binary variable change (burnt versus not burnt) based on multiple predictor variables, however the latter approach was not adopted. The algorithm was developed and tested using SPOT Vegetation ten-day (S10) syntheses covering Canada for the periods 21st April to the 10th of October 1998 and 1999. Composites were corrected for atmospheric effects and normalized to a common viewing geometry using a kernel-based bi-directional reflectance distribution function (BRDF) model. Training samples covering a wide range of boreal forest environments were selected across Canada representing 1998 burnt forest (163 polygons, 2504 SPOT Vegetation pixels) and non-burnt forest (222 polygons, 2821 SPOT Vegetation pixels). Predictor variables consisted of ten day and surrounding thirty day changes in reflectance (red (B2), NIR (B3), and MIR) and in two vegetation indices, each of which were normalized to the background reflectance trajectory to control for seasonal changes in vegetation phenology. A model containing four change metrics was selected based on backward stepwise logistic regression ($R^2 = 0.77$, $p < 0.001$). Examining accuracy assessment curves showing error rates in the training data over a range of probabilities derived an optimal decision threshold for the model. Error rates were adjusted to account for the *a priori* probability of burning at a continental scale considering that a given rate of commission error (falsely detected change) leads to a significantly larger area of incorrectly classified pixels compared to the same level of omission error. Spatial-contextual tests were developed to follow the per-pixel logistic model in an attempt to further remove noise and increase the sensitivity of detection. These tests involved filtering small clusters (less than five pixels) of pixels labelled as burnt, followed by iterative region growing from the high probability burnt pixels to adjacent, lower probability pixels. As an alternative to removing small clusters, it was demonstrated that active fire locations identified using satellite data, if available, are highly effective in removing change pixels unrelated to fire activity.

An initial validation of the algorithm was performed by applying the model to 1998 and 1999 SPOT Vegetation S10 data covering Canada and comparing results to fire surveys and burnt area statistics from provincial forest fire management agencies. Errors of commission for 1998 were found to be small (approximately 5% in Alberta), and most burns larger than 10 km^2 were accurately detected and mapped (R^2 = 0.97; RMS = 3.3 km^2 in Alberta). SPOT Vegetation burnt area estimates for Canada were smaller by 23% in 1998 and 11% in 1999 by comparison to agency statistics compiled by the Canadian Forest Service. In several northerly locations burnt areas were underestimated due to cloud artefacts over dark burns resulting from the maximum NDVI compositing criterion used to produce the S10 products (Fraser *et al*. 2002). For the year 2000 dataset, S1 products were processed between the 21st April and 30th September 2000 for windows C1 and C2 and between the 21st April and 10th October 2000 for the US window.

For the implementation of the algorithm at the JRC some modifications to the algorithm were required. After making extensive enquiries into the availability of software for making BRDF corrections to SPOT Vegetation S1 data, the author was made aware with private communications from experts in this field that making these corrections to S1 data would, in fact, decrease the quality of the data. Also, implementation of the BRDF correction would be very difficult within the time constraints of the project. These concerns were reported back to the CCRS group who endeavoured to develop an algorithm yielding comparable results but without the need for BRDF corrections to be applied. Modifications were made and a satisfactory burnt area algorithm that utilised non-BRDF corrected products was delivered.

9.1 CCRS pre-processing module

The only pre-processing requirement of the CCRS burnt area algorithm is the removal of non-burnable pixels according to the UMD land cover mask. This is to ensure that water and non-vegetated land surface areas are removed in a consistent manner (like in other processing windows). The c-shell script gba_preprocessor.csh is used (Chapter 3). An example of the command used to pre-process the data for all of those dates indicated in the text file input_dates.txt is:

gba_preprocessor.csh C1 0 - 0 - 0 - - - - 0 - - - 1 burn 1

9.2 CCRS compositing procedure

The pre-processed SPOT Vegetation data was composited over a time period of ten days according to a maximum NDVI criteria (Section 4.1). The text file input_dates.txt was edited to reflect the time period to be composited. To make the composites, the c-shell script gba_min_nir_composite.csh was used. This also yielded composites of the NDVI and SWVI according to the maxNDVI criteria. An example of the command used is:

gba_max_ndvi_composite.csh C1 ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin burn d

9.3 CCRS burnt area algorithm procedure

9.3.1 Implementation of the CCRS algorithm

The code for the CCRS algorithm was developed mainly in the ARC Macro Language (AML) that utilises commands and routines applicable to binary arrays. For these programs to be used, an ARC-INFO license is required. The program used both ARC and ARC-GRID commands. Other data manipulations are undertaken outside of the AML by generic c programs and c-shell scripts. A series of AML programs are run in sequence, controlled by a c-shell script that requires only a small amount of user intervention.

The AML programs operate on datasets in ARC-GRID format only. Therefore, one of the first stages in deriving the burnt area products is the creation of the input data products in GRID format. This is achieved using the c-shell script gba_ccrs_preprocessor.csh with the following input commands:

gba_ccrs_preprocessor.csh C1 burn /home1/arcinfo721/arcexe72/bin

where, acronym indicates the region being processed (e.g. C1). The level of pre-composite processing (i.e. burn level, indicating that non-vegetated pixels have been removed) is also given at the command line followed by the full path to the ARC commands. The program is run within the working directory and utilizes ARC commands imagegrid and floatgrid. A complete description of the c-shell script is given in Annex I.

The program creates, in a directory called ccrs_grids, GRID files for the following products: bands B2, B3 and MIR (imported as integers) and indices NDVI and SWVI (imported as floats) after conversion using the generic c program short2float.c (see Annex A). Prior to the

creation of the GRID images, the unique region (non-buffered) within the sub-window is extracted.

An important component of conversion to GRID format is the presence of a header file that can be interpreted by ARC-INFO. An example is shown below that illustrates the information required

```
nrows 1380
ncols 13400
nbands 1
nbits 16
layout bsq
ulxmap -179.8169647
ulymap 74.81696471
xdim 0.0089285714
ydim 0.0089285714
```

where, nrows and ncols indicate the geometry of the image, nbits equal the number of bits per pixel (integer) and ulxmap, ulymap and x(y)dim indicate the map position of the centre of the upper-left position and pixel spacing (decimal degrees) respectively. Please note that for floating point images the specifics of the header file change. Consult the manual pages of the ARC-INFO system for further instructions.

9.3.2 Other Input Data

The CCRS burnt area algorithm requires the following datasets as input data. Input channels for each ten day composite period include red (B2) reflectance (scaled from 0-2000), middle-infrared (MIR) reflectance (0-2000), NDVI (scaled from –100 to 100) and SWVI (scaled from –100 to 100). These files are initially located in directories created by the compositing programs. After application of the c-shell script gba_ccrs_preprocessor.csh these are now in GRID format.

In addition to these files, a water mask file must be present. This image is created from land cover maps provided by CCRS. An ecozone image that defines similar climatic and ecological zones must also be present. This product is used to normalise the information extracted from the satellite data to take into account, for example, seasonal changes due to phenology. Background vegetation groupings were defined based on nine land cover types derived from a SPOT Vegetation classification stratified over thirteen Canadian terrestrial ecozones, yielding a total of 117 background vegetation groupings. The purpose of the additional ecozone stratification is to account for north-south phasing differences in the

timing of the growing season. For the ecozone product of the USA, this product was from derived using information from the IGBP land cover product and pre-defined North American ecozones. An illustration of the vegetation cluster image for Canada is shown in Figure 24. These files must be co-registered with and have identical geometry to the window of interest. In addition, these files must have the naming convention $acronym_eco.bsq and $acronym_water.bsq and be accompanied by ARC-INFO format header files. These files are located within a directory called common, situated in the working directory (i.e. ./C1/common).

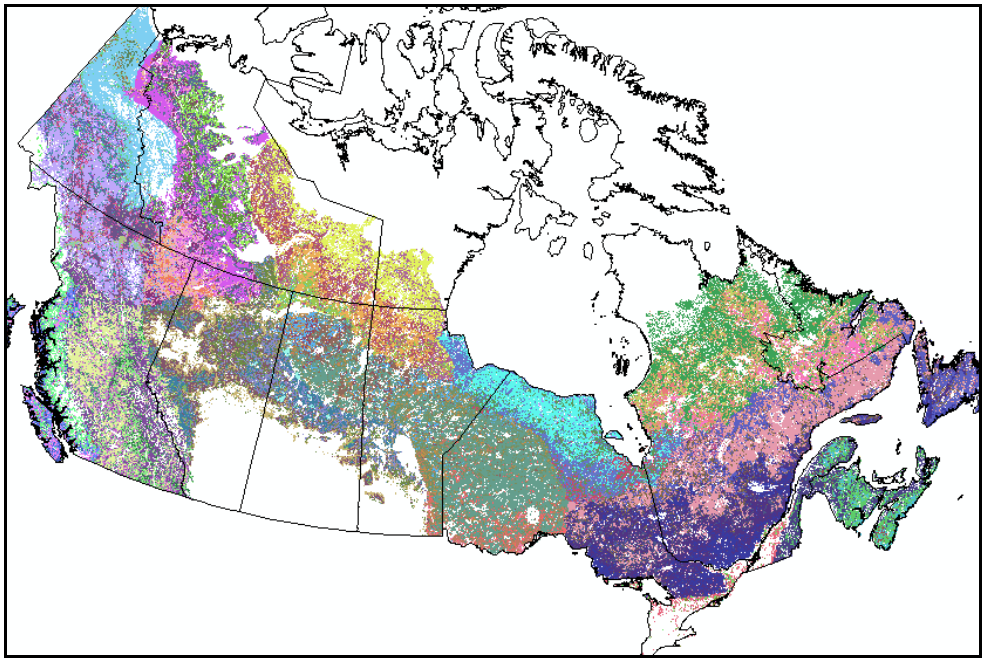


Figure 24: Nine land cover types from a SPOT Vegetation classification were intersected with thirteen terrestrial ecozones to produce 117 vegetation clusters. Each pixel's trajectory was normalized according to the trajectory of its cluster.

9.3.3 CCRS algorithm: stage 1

The complete implementation of the CCRS burnt area algorithm is achieved using the c-shell script gba_ccrs_processor.csh with the following input commands:

gba_ccrs_processor.csh c1 /home1/arcinfo721/arcexe72/bin

where, acronym indicates the region being processed (e.g. c1). Note that acronym should be given in lower case letters so that the GRID programs work. The full path to the ARC commands is also given at the command line. A complete description is given in Annex I.

The first set of programs that this c-shell script activates, recodes and converts into GRID format the ecozone and water masks situated in the directory named common. An AML script is automatically run within GRID called recode_masks.aml (see Annex I). This step is necessary to recode the values of the binary products. The next stage is the creation of a cloud mask product. Pixels were flagged as contaminated if the red reflectance (B2) was greater than three standard deviations from the mean growing season reflectance of their respective vegetation grouping. A maximum 7% red reflectance was enforced for all pixels to control for the high variation observed in mountainous areas of Canada. These red reflectance's were extracted from six image values between the 1st-10th July and 21st – 31st August 2000 composites. The AML program make_cloud_thres.aml is automatically run to create this cloud mask product (see Annex I). All of these GRID products are stored in the directory named common.

9.3.4 CCRS algorithm: stage 2

The second stage performs the cloud screening and sets up the initial statistics for the background vegetation clusters of the first composite. An AML script is run within the directory of the first composite period (i.e. 20000421). This script is called baabz_setup1_nobrdf.aml (Annex I). Once this is completed, a second set up script is run in the directory named common that initialises various common grids that will be used in the next stages of processing. This script is called baabz_setup2_nobrdf.aml (Annex I).

9.3.5 CCRS algorithm: stage 3

The third stage undertakes the main component of the burnt area algorithm procedure. A logistic model is applied separately to consecutive ten-day periods producing an output value ranging from zero to one that represents the probability of a pixel being burnt during that period. An AML script applies the regression model to the series of ten-day composites, a maximum probability value is accumulated for each pixel to yield a product indicating the highest probability of burning during the fire season. A separate grid indicates the ten-day period during which this highest probability occurred. The probability of burning for a pixel during each ten day period is given by the following multiple logistic regression model:

$$P(burnt) = 1 / [1+ EXP -(-1.51 + 0.19\times\Delta MIR_{30} + 0.16\times\Delta NDVI_{10} + 0.064\times\Delta Red_{10} + 0.015\times\Delta MIR_{30} - 0.01\times NIR)$$

where, $\Delta SWVI_{30}$ equals the 30-day change in a short-wave based vegetation index (SWVI) surrounding the current ten day interval (i.e. $\Delta SWVI10_{t-1} + \Delta SWVI10_{t} + \Delta SWVI_{t+1}$), $\Delta NDVI_{10}$ equals the ten day change in NDVI, ΔRed_{10} equals ten day change in red (B2) reflectance, ΔMIR_{30} equals the 30-day change in MIR reflectance and NIR equals the near-infrared (B3) reflectance for the current ten day period.

Two special features of the change metrics must be considered during their computation. First, each pixel's ten and thirty-day change is normalized to the corresponding change occurring in similar background vegetation (see Figure 25). This normalization accounts for reflectance changes attributable to seasonal vegetation phenology, especially during the fall senescence period when multi-temporal algorithms will be susceptible to producing commission error. The ecozone GRID is utilised for this operation. This normalization can be expressed for the middle-infrared (MIR) as:

$$\delta MIR_{10} = \Delta MIR_{10} - \Delta MIR_{10b}$$

where, b equals the average background vegetation value.

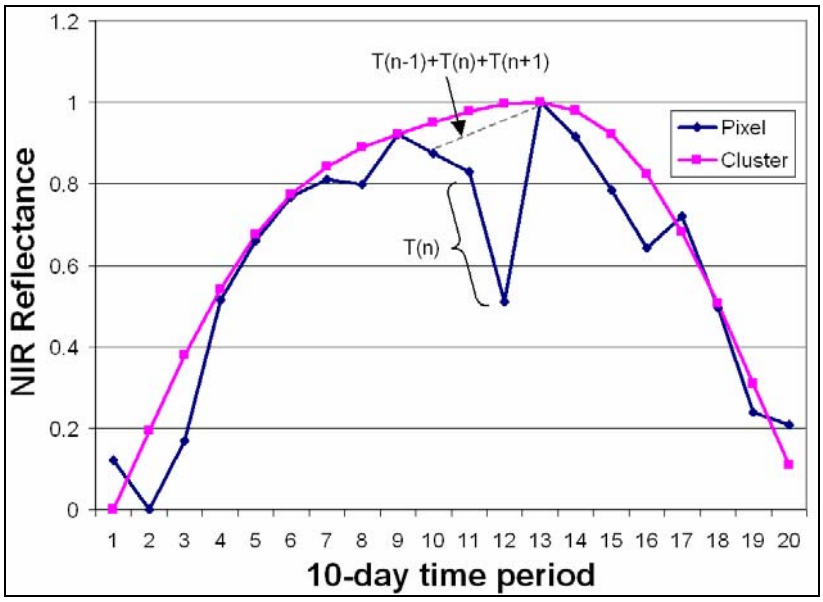


Figure 25: Hypothetical NIR reflectance trajectories of a single pixel (dark blue) illustrating how ten and thirty day change metrics are computed. The thirty-day metric is useful for flagging single-date changes in reflectance caused by factors other than fire (e.g. atmosphere, extreme viewing geometry). The metrics are normalized for phenological variation according to the trajectory of background vegetation (purple).

A second feature of the change metrics is that any pixels contaminated by snow or cloud are skipped. For each of the composites, the CCRS algorithm is applied and the cumulative probabilities calculated using the AML program baabz_logistic_nobrdf.aml, operated by the AML program baabz_go_nobrdf.aml (or baabz_go_usa_nobrdf.aml for the US window, because of an extra ten day composite in October being present). Both programs are described in Annex I. Each time a composite is processed, the program calculates the logistic regression probability and updates the cumulative maximum burn probability product and change date product. An example of the cumulative probability of likelihood of burning based on applying the regression model to data from the 1998 fire season is shown in Figure 26. The regression model is applied to all of the composites (excluding the first composite at the end of April).

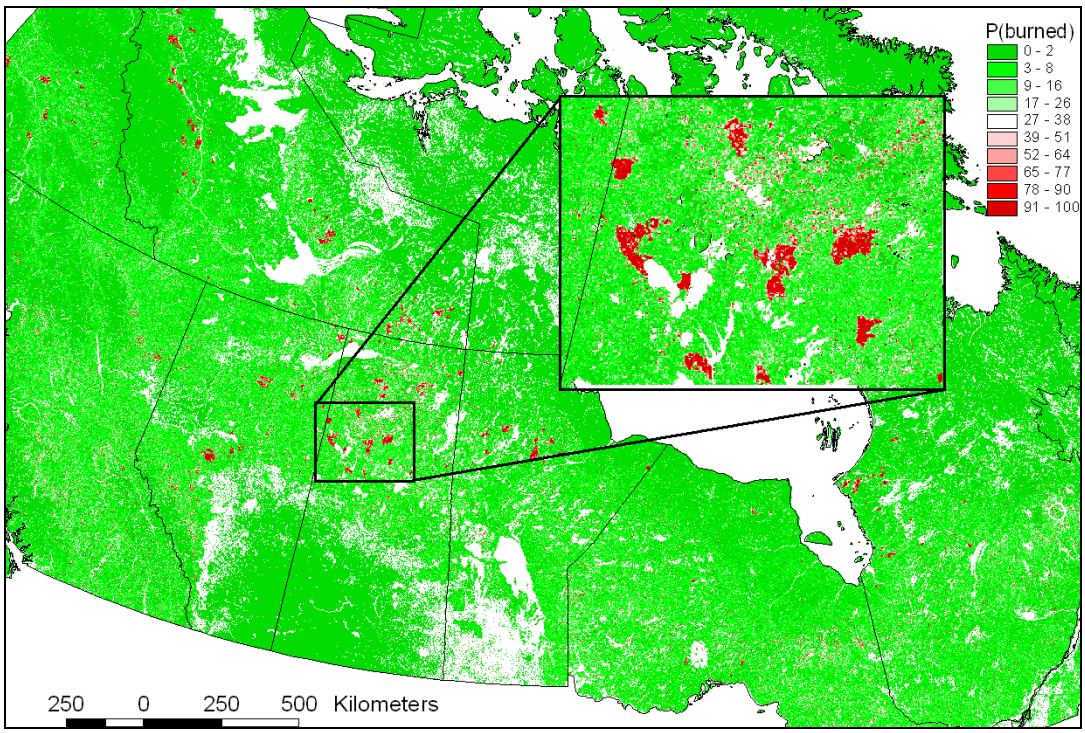


Figure 26: Maximum probability of burning for each 1 km^2 pixel in Canada based on applying a logistic regression model to a series of ten-day SPOT Vegetation composites for the 1998 forest fire season.

9.3.6 CCRS algorithm: stage 4

In the fourth processing stage, the cumulative probability map is converted to a binary burnt area mask by applying a cut-off threshold where all pixels having a probability greater than

0.97 are labelled as burnt. This decision threshold was optimised to balance the predicted area of commission and omission error across Canada (Figure 27).

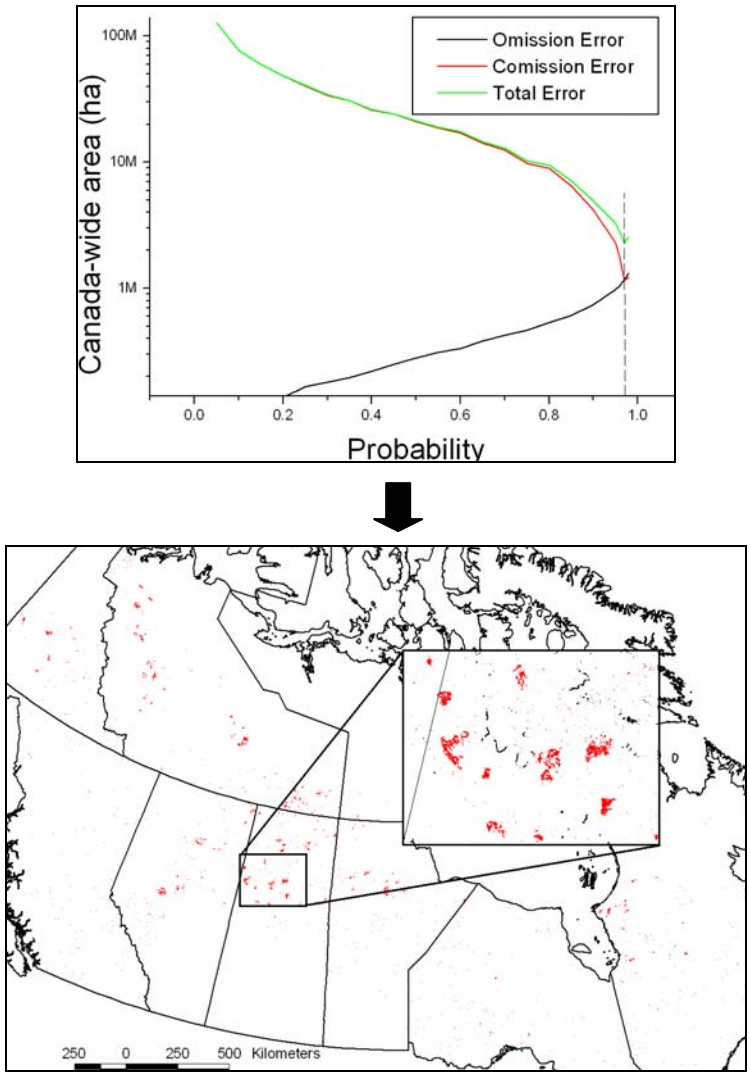


Figure 27: Accuracy assessment curves showing predicted area of commission and omission errors over a range of probability levels were used to derive an optimal decision threshold (0.97) for producing an interim burnt area mask.

The next step involves several contextual/spatial tests. Initially, burnt pixels lying adjacent (using four way connectivity) to major water bodies are filtered. These pixels are susceptible to producing false burns due to slight multi-temporal misregistration. Then all contiguous burnt area clusters smaller than five pixels are assumed to represent noise and are removed. This test removes a significant amount of commission error, yet eliminates only a very small proportion of the real burnt area since boreal burns smaller than 200 hectares account for only about 2% of burnt area in Canada (Stocks, 1991). The next contextual step involves iterative region growing from the remaining burnt pixels (> 0.97 probability) to adjacent pixels having

lower probability (> 0.35 probability). The purpose of this step is to recapture most of the burnt pixels that were missed by using the conservative 0.97 decision threshold. Region growing continues until all burnt patches in the window stop increasing in size. The last step removes any burnt area clusters containing fewer than 15% of the high probability (> 0.97) pixels. In some instances a small cluster of false burnt area pixels can grow to a large area of pixels having probability > 0.35. In almost all cases, real burns comprise at least 15% of the high probability pixels. An illustration of the spatial/contextual measures applied to the cumulative (with the threshold applied) probability product is shown in Figure 28.

(a) (b) (c)

Figure 28: Illustration of spatial/contextual steps used to produce final burnt area mask. (a) shows the logistic regression classification with a threshold of 0.97 applied. (b) shows the classification after the removal of small clusters and (c) shows the classification after region growing to lower probability pixels.

The final stage of processing is implemented using the AML program baabz_burnmap_nobrdf.aml (Annex I).

The final burnt area binary and corresponding date of burn product is written out as generic binary images (as opposed to GRID format). Recoding of the binary product is necessary to ensure that the image is written out in byte format (one byte per pixel). The likely date of burn for each pixel determined as being burnt is written to file as integer format. These final (non-GRID format) products are written to a directory named ccrs_ba within the working directory and ENVI headers are created for the files. A clean up of the data is then performed. From the burnt area product indicating the estimated timing of the burnt areas, monthly burnt area products are derived. This was achieved using standard masking procedures available on desktop image processing software or the programs described in Chapter 13.

10 CNR algorithm module

The CNR group, comprising Pietro Alessandro Brivio, Elisabetta Binaghi, Ignazio Gallo, Claudia Giradino and Marta Maggi were responsible for the delivery of a burnt area algorithm for central Africa. The algorithm was developed over a fixed region of the Africa continent north of the equator up to 18 degrees N and between –18 and 52 degrees E. The time period of study for this region was from the 1st December 1999 to the 31st of March 2000 and again from the 1st of October 2000 until the 31st December 2000.

10.1 CNR methodology for burnt area detection

Approaches to burnt area mapping range from the application of multiple tests on spectral values or derived indices to temporal analysis of remotely sensed data and to traditional image segmentation techniques. However to date, a global burnt area mapping algorithm has not been developed, although there is a general agreement that multi-temporal algorithms have the greatest potential for operational implementation. The description of the methodology given here has been largely extracted from Brivio *et al*. (2001a). Further information about this algorithm can be found in the following articles (Brivio *et al*., 2001b; Brivio *et al*., in press; Brivio *et al*., submitted to the IJRS).

10.1.1 Neural network approach

The approach proposed by the CNR group is a contextual classification strategy aimed at exploiting spatial and temporal information required for identifying burnt areas in low resolution imagery such as SPOT Vegetation data. The classification is based on Multi-Layer Perceptron (MLP) (Paola and Schowengerdt, 1995) which is used as a soft classifier considering the gradual activation values of the output neurons as final partial membership to classes. The use of the MLP model is motivated by the experimentally proven effectiveness in running as a soft classifier (Binaghi *et al*., 1999) and the well-documented capability in dealing with patterns described by complex features that in our case result from the modelisation of spatial and temporal context. Figure 29 shows the overall classification strategy receiving as input a time series of imagery from day d_j to day d_{j+k} and producing in output a burnt area map M_j associated to the day d_j. The value of five was selected to

represent k, to account for differences in radiance values received at the sensor due to the variation in the observation angles.

The MLP model is inserted within a two stage classification strategy aimed to detect burnt pixels in daily images. The approach of processing daily images was preferred to the compositing procedure used widely in the analysis of a set of multi-temporal images acquired by wide field of view polar orbiting sensors (Cihlar *et al*., 1994). Compositing is a form of data fusion, that selects from a series of collocated pixels of different orbits the pixel that best satisfies some criteria allowing to obtain a single ideal dataset with reduced atmospheric and cloud contamination and angular effects. However, these procedures that select the most suitable pixel form a set of measurements taken at different dates, often produce strong radiometric artefacts when compositing individual channels (Brems *et al*., 2000). The use of daily images preserves the full range of information contained in the original radiometric measurements.

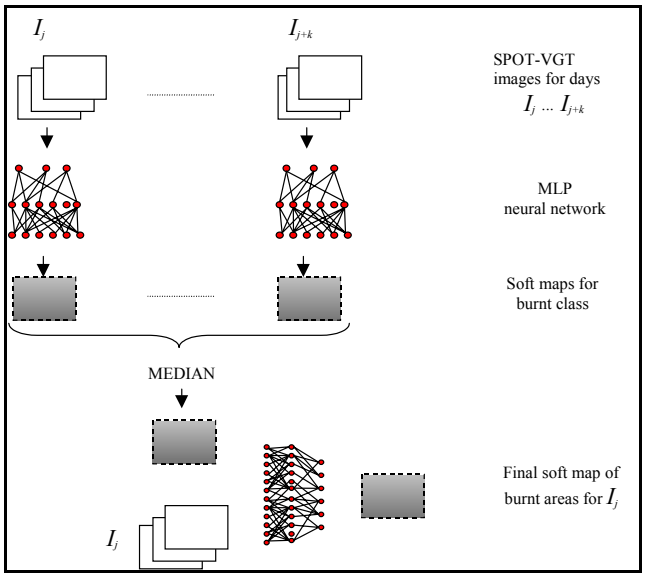


Figure 29: Diagram showing the classification strategy used for burnt area mapping in SPOT Vegetation time series of data.

10.1.2 Network training

In any supervised classification procedure the training phase is an important step. In order to optimise the selection of training samples to be used for burnt area detection the effect of the sun-target-sensor geometry on the spectral signature of burnt areas was estimated. In fact,

satellite sensors acquiring data at a global scale are affected by bi-directional effects due to their large swaths. These effects cause a change in the spectral features of the land surfaces according to the sun-target-sensor geometries (Shepherd and Dymond, 2000). In other words, the observed reflectance varies with the angles at which the pixels are illuminated and viewed, that is most surface objects exhibit anisotropic reflectance properties (Hu *et al.*, 2000). This behaviour is described by the bidirectional reflectance distribution function, or BRDF. For this reason, the BRDF of different surfaces has been estimated using the semi-empirical approach provided by the, algorithm for MODIS bidirectional reflectance anisotropies of the land surface (AMBRALS) developed by Wanner *et al.* (1995). Figure 30 represents a typical spectral profile of a burnt area in a savanna environment compared to the profile of an unburnt surface within the same vegetation cover. The reflectances refer to the four channels of the SPOT Vegetation sensor.

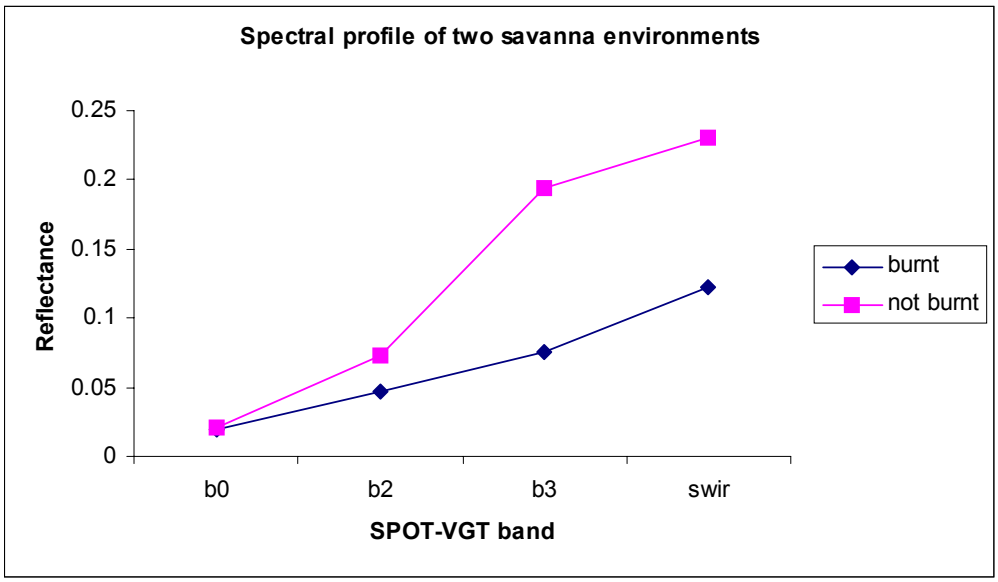


Figure 30: Spectral profiles of a savanna environment before (not burnt) and after (burnt) a fire event.

In the visible bands, the burnt surface has values not too different from the unburnt one. On the contrary, the NIR (B3) and MIR reflectances show a strong decrease compared to the undisturbed condition. The near-infrared band is the most sensitive to the fire event. For the present analysis, a target area representing a homogenous, recently burnt surface in a savanna environment was chosen. For the estimation of the BRDF of this target area, a short time-series of SPOT Vegetation daily images acquired in December 1999 was considered. The dataset included 18 measurements of atmospherically corrected reflectances of the same

burnt area of the four SPOT Vegetation channels and of the four illumination/observation angles, namely the satellite viewing zenith angle (VZA), satellite viewing azimuth angle (VAA), solar zenith angle (SZA) and solar azimuth angle (SAA). Within this dataset the values of SZA were between 35 and 43 degrees, and values of VZA between 2 and 56 degrees.

The period of analysis allowed the CNR group to assume that the spectral characteristics of the target surface are stable, even if *in situ* measurements in a Namibia savanna site made by Trigg and Flasse (2000) indicated that the blue channel (B0) is strongly affected by atmospheric conditions and the MIR reflectance values return to pre-burn conditions a few days after the fire event. Assuming that the analysed surface and residual atmospheric effects are constant in time, the geometry of the sun-target-satellite system can be considered as the most important cause of the variability in daily reflectance values. This variability is represented in Figure 31. It is interesting to note the presence of a regular five-day cycle within the period considered. Moreover, periodic no-data values are present, due to the partial daily coverage of the system at the latitude of the study area.

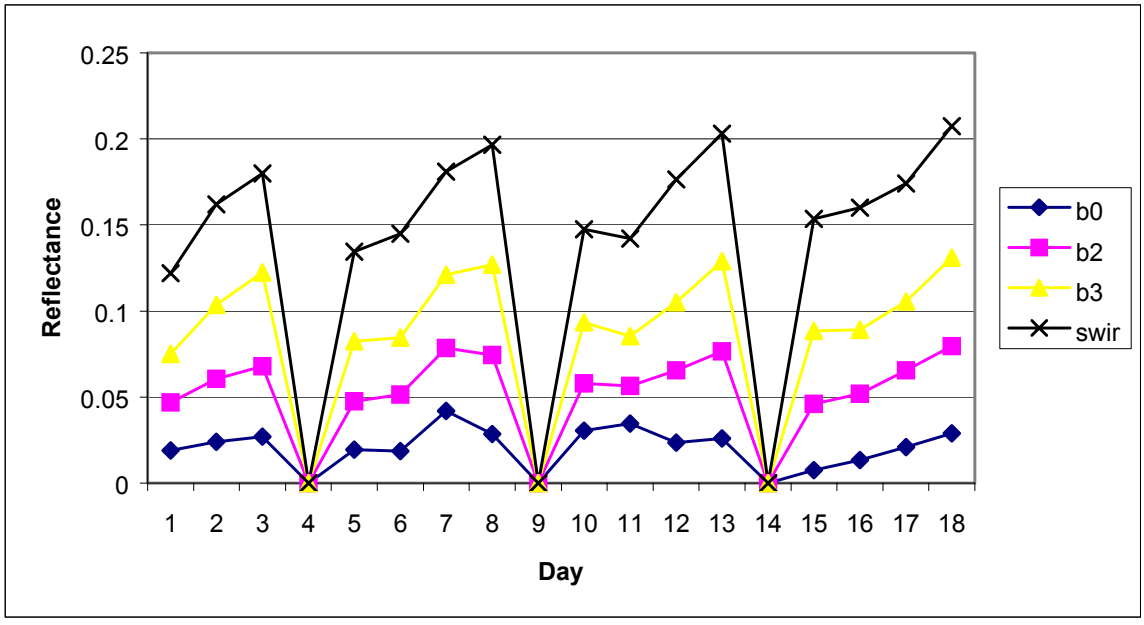


Figure 31: Effects of the observation angles on daily reflectance values in the four spectral SPOT Vegetation channels as observed from 1st to 18th December 1999.

10.1.3 The AMBRALS model

The AMBRALS RossThick-LiSparse kernel driven model (Wanner *et al*., 1995) was used to model the BRDF as a weighted sum of a volume scattering function and a surface scattering function and a constant. In this study, AMBRALS was used to model the reflectance of the burnt pixels under several illumination/viewing conditions. This allowed the developers to establish if training samples of burnt area must include pixels characterised by as large as possible number of illumination/viewing conditions or, on the contrary, not constrained by large variations of illumination/viewing conditions. The latter, occurring for near-Lambertian surfaces, would permit a faster and easier approach in the training sample collection for the burnt area classification. A complete description of the results of these experiments can be found in the report made by the group of CNR (Brivio *et al*., 2001a).

To summarise, the results show that when the sun is very high above the horizon, the bidirectional reflectance in the four SPOT Vegetation channels is quite independent by the viewing zenith angles and the variations are limited within ± 2.5%. To the contrary, when the sun zenith becomes high, the bidirectional reflectance of a burnt area changes with the viewing observation geometries ten times more than in the previous case. This may constrain the selection of the training samples of burnt pixels to be used in classification algorithms. If images are acquired at low sun zenith angles, the training phase may be accomplished quite quickly because the selection is not constrained by the necessity to have burnt areas observed at several viewing angles. On the other hand, if SPOT Vegetation data are acquired at high sun zenith angles, the training samples should include pixels spread all over the image, due to the relevant changes of surface spectral properties with the viewing zenith angles. By adopting this method, we are assured to build-up a more representative training set to be used in supervised classification algorithms for burnt area detection.

10.1.4 Stage 1 of the classification approach

In the first stage of the classification process only the spectral information is taken into consideration. Each image of the time series (I_j , … , I_{j+k}) is classified separately by the MLP according to a pixel-by-pixel classification scheme shown in Figure 32. The following six land cover classes were identified:

Class 0: Water

Class 1: Burnt

Class 2: Not Burnt

Class 3: Doubtful Burnt

Class 4: Cloud Shadows

Class 6: Clouds

About 25,000 pixels were used to train the neural network and about 12,000 pixels were used to verify its generalisation capability. These samples were collected on images of five consecutive days of the SPOT Vegetation cycle at the beginning of March 2000. The MLP neural classifier receives input spectral values (from bands B2, B3, MIR) and computes the gradual membership of that pixel to all the six classes conceived. Six, soft maps constitute the result of this stage of the classification for each daily image processed indicating the gradual membership of pixels to the corresponding classes.

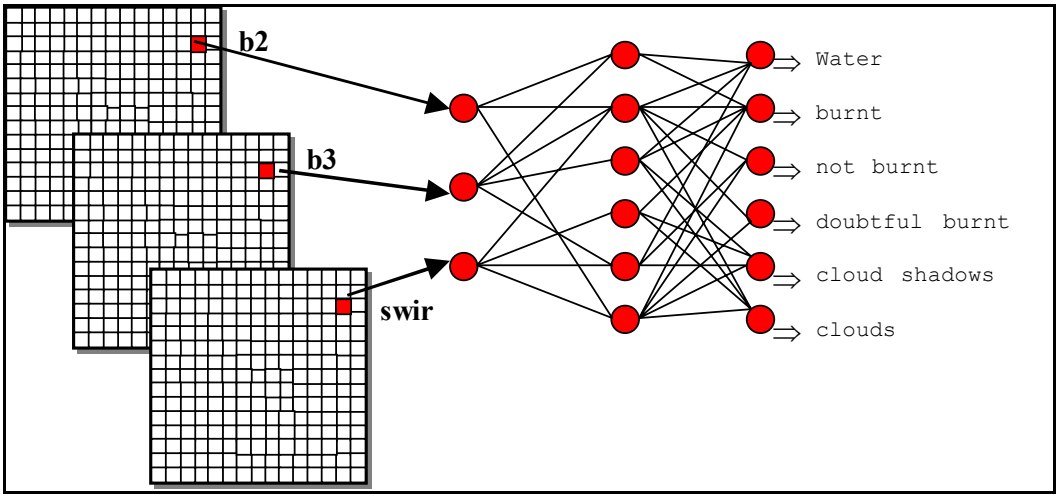


Figure 32: The MLP neural network used in the first phase of the classification of SPOT Vegetation daily images for burnt area mapping.

10.1.5 Stage 2 of the classification approach

The second stage of the classification process is aimed to produce the final classification of the daily image I_j. Whilst most of the approaches reported in the literature exploit multi-temporal information using change detection, the proposed approach is based on the idea that the burnt surface signal persists for some time after the fire events. The length of time that burnt area signals exist after a fire is highly dependent on physical evolution of the post-burn surface in the particular ecosystem, and on the spectral resolution of the detecting satellite.

In this framework, the series of K soft maps related to burnt class are considered and fused together by applying the median operator, pixel-by-pixel. Values in the resulting median map synthesize information about the temporal behaviour of the pixels and play the role of reinforcing or attenuating the strength of membership to burnt class in the final map. Variability in spectral response due to no data conditions or clouds is automatically managed within this data fusion pre-processing stage contributing to make the classification output more reliable. The second stage neural classifier is configured to receive in input I_j pixel spectral values and values of a 3x3 window drawn from the median map and centred on the coordinates of the pixel to be classified (as shown in Figure 33).

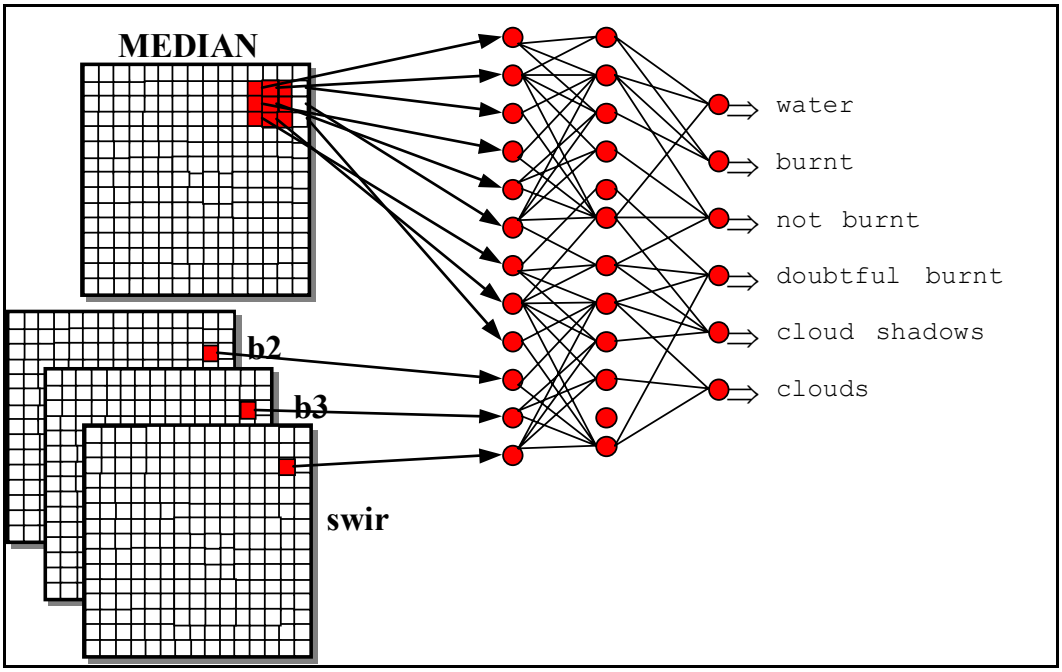


Figure 33: Diagram showing the MLP network architecture used for the final classification stage.

10.2 Implementation of the CNR algorithm

The CNR algorithm including the neural network component was coded in Visual C++ originally for operation on a PC. Code has also been developed to operate under LINUX and UNIX environments. Contact should be made with the algorithm developers at CNR.

11 UOE Algorithm Module

A joint Portuguese and Brazilian contribution, comprising of Adélia Sousa (University of Évora, Portugal), José Pereira (UTL, Portugal), Ana Cabral (UTL, Portugal) and Alberto Setzer (CPTEC, National Institute for Space Research, Brazil) were responsible for producing an algorithm for the detection of burnt areas from SPOT Vegetation data over the Legal Amazon during the year 2000 (Silva *et al*., 2002b). The development of the algorithm was made using SPOT Vegetation S1 data covering the region shown in Figure 34 for the months of June to October 2000.

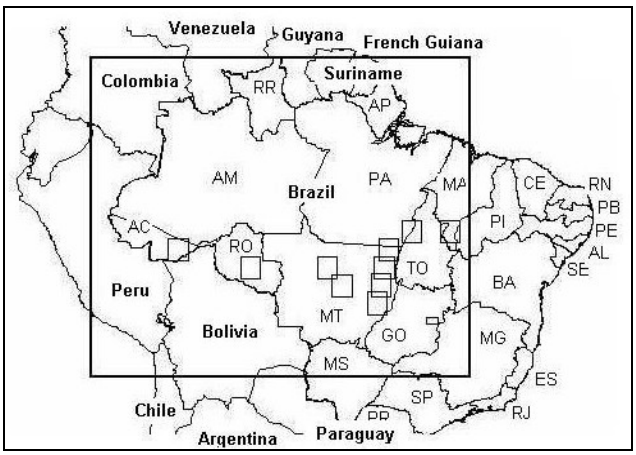
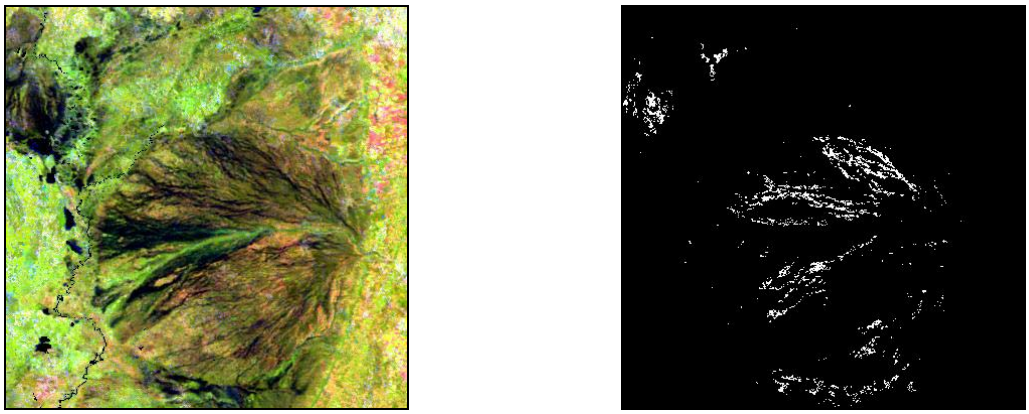


Figure 34: The Brazil data window used by colleagues at UOE/UTL to develop the burnt area algorithm. The small boxes indicate the location of ten Landsat TM images that were used for validating the algorithm's performance.

The original algorithm developed by this group and reported in a conference proceedings (Silva *et al*., 2002b) had several shortcomings with reference to the GBA2000 product. The shortcomings were:

- The algorithm used composite images for three consecutive monthly periods. This meant that a monthly burnt area product for December 2000 was impossible to produce, because no data were available for January 2001.
- The algorithm when applied to monthly composite data outside the time period from which the algorithm was derived, performed badly. This was due to the false detection of surfaces flooded for a time period of greater than one month. The use of geographical and temporal masks to remove these false detections wanted to be avoided if at all possible. This problem can be observed in Figure 35. In the left image, the monthly composite for March 2000 is shown for a region in southern

Brazil as an RGB image (Red = MIR composite band, G reen = B3 composite band, Blue = B2 composite band). The dark areas in the image are regions of flooding. When the burnt area algorithm is applied, the burnt area map shown in the right image is derived. We need to account for the occurrence of flooding because the region burns during the dry season.

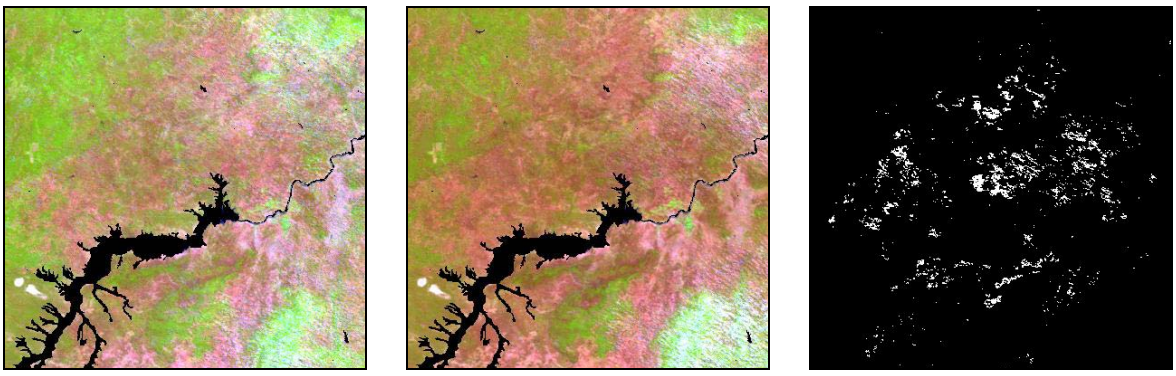


(a) (b)

Figure 35: An illustration of the occurrence of flooding in southern Brazil during the wet season. These are displayed in dark colours in image (a). Image (b) shows the false detections of burnt areas because of this flooding.

- The algorithm indicated that there was an incredible amount of burning activity (and therefore burnt areas) occurring in every month in the mountains and foothills of the Andes, in particular south of the equator. The authors of the algorithm did not believe that all of these were true burnt areas but rather due to a number of factors including snow melt, the presence of sun shadow caused by strong topographical changes and phenology. The authors proposed an extremely coarse mountain mask that was considered unacceptable for the GBA2000 project.
- The author of this report wanted to apply the algorithm over the region of interest called BR (see Figure 3). The window BR was larger than the test window used by UOE and included also the eastern part of Brazil. When the algorithm was applied to the window BR, large areas of false detections (confirmed from Landsat TM quicklooks) were indicated in this region. The region is characterised by dry, thorny scrub and shrublands, that becomes dry very quickly, but at different stages during the dry season. This problem can be observed in Figure 36. In the left image, the monthly composite for May 2000 is shown for a region in eastern Brazil. In the centre image, the composite for June 2000 is shown. Both are displayed as RGB images (Red =

MIR composite band, Green = B3 composite band, Blue = B2 composite band). When the burnt area algorithm is applied, the burnt area map shown in the right image is derived. The areas indicated as being burnt have undergone significant change in the time between the two images (between one and two months). A closer look at these areas using Landsat TM quicklook data showed very little evidence of burning at this time. The changes were attributed to drying out of the vegetation.



(a) (b) (c)

Figure 36: An illustration of the occurrence of vegetation drying out in eastern Brazil during May (a) and June (b) 2000. Image (c) shows the false detections because of this drying out.

The original algorithm was implemented to derive monthly, burnt area maps for the months of January to November 2000. This was done so that comparisons between the original and modified products could be made, especially as the accuracies of the original products in the areas where Landsat TM data were available were high (Silva *et al*., 2002b). The algorithm is based on a classification trees approach using CART software. The algorithm was trained with monthly composited data selected from certain training sites. To solve some of the shortcomings highlighted above, data were also provided to the UOE team where flooding was evident and also from the region of eastern Brazil. In addition, the UOE group were asked to derive a set of decision rules that utilised only pre- and post-burn sets of data (i.e. two monthly composite images) so that a burnt area product could be derived for December 2000. Solutions to reduce false detections in the Andes Mountains were investigated.

11.1 UOE pre-processing module

The pre-processing requirements of the UOE burnt area algorithm for Brazil are the removal of data acquired at extreme viewing zenith angles, those contaminated in the MIR channel and those pixels characterised by non-vegetated surfaces. The c-shell script

gba_preprocessor.csh is used (Chapter 3 and Annex C) because all of the above requirements can be satisfied. An example of the command used to pre-process the data for all of those dates indicated in the text file input_dates.txt is:

gba_preprocessor.csh BR 1 60 1 1000 0 - - - - - 0 - - 1 basic 1

For each day, the pre-processed data would contain data that were masked for saturation in the MIR channel (all pixels with a DN greater than 1000), non-vegetated and water surfaces and pixel data acquired at extreme viewing zeniths (greater than 60 degrees).

11.2 UOE compositing procedure

The pre-processed SPOT Vegetation data was composited over a time period of one month according to a third lowest minimum near-infrared (NIR) criteria (Section 4.1). The reason for the selection of the third lowest NIR value is that cloud shadows tend to be similar to, or darker than burned surfaces at all wavelengths. However, they are very ephemeras features. Empirical observations showed that it was very unlikely for any given pixel to be cloud-shaded more than three times in a month. If a pixel appears dark at least three times in a month, that pixel is considered as being a potential burn. The method eliminates all pixels that were in shade once or even twice in any given month. The implication is that the burnt area is likely to be detected not on the day it occurs, nor on the following day, but only on the third day (assuming it is darkest the day it occurs, and gradually loses the charcoal layer). Given the post-fire spectral dynamics, even in tropical savannas, this 2-day detection delay appears to be perfectly acceptable (Cabral *et al*., accepted). The text file input_dates.txt was edited to reflect the time period to be composited. To make the composites the c-shell script gba_value_nir_composite.csh was used (Section 4.3). An example of the command used is:

gba_value_nir_composite.csh BR ~tanseke/src/idl /mtvdata/mm-rsi/envi_3.4/idl_5.4/bin basic m 3

This command indicates that level two products (including, for each pixel, the number of cloud free days, the day in the time period from which the pixel was selected for the composite and the status image) were produced from fully pre-processed data over a monthly time period with the selection of the third lowest near-infrared value.

11.3 UOE burnt area algorithm procedure

The burnt area algorithm has been developed by means of a supervised classification technique based on the Classification and Regression Trees (CART) theory (Breiman *et al*., 1984). Training sets were used to derive a decision tree from which a set of classification

rules (i.e. spectral thresholds) were extracted. The threshold values are applied to both monthly composites (times 1 and 2) in the red (B2), near-infrared (B3) and middle-infrared (MIR) channels and the difference between the composite values in the near-infrared (B3) channel. The algorithm is applied only to those pixels that are clear in both composites. The output burnt area map is a classification of the composite image at time 2 into burnt or not burnt classes. The classification rules to determine a burnt surface are summarised in Table 3. Those pixels that satisfied rules two, five or nine are shown as burnt in the monthly product.

Table 3: The classification rules for the burnt surface class. A pixel is classified as burnt if it satisfies the conditions of classification rules two, five or nine, otherwise the pixel is not burnt. The variables used are the SPOT Vegetation red (B2), NIR (B3) and middle-infrared (MIR) channels. ΔijVARIABLE = VARIABLE $_{time\ i}$ – VARIABLE $_{time\ j}$, where time I and j refer to the composite images at times 1 and 2 respectively. The values shown are in DN.

Rule	Conditions (AND)
1	$Nir_2 \leq 230.5$; $mir_1 \leq 336.5$; $mir_2 \leq 199.5$ then 0, else
2	$Nir_2 \leq 230.5$; $mir_1 \leq 336.5$; $mir_2 > 199.5$; $\Delta_{21}(nir) \leq -83.5$ then 1, else
3	$Nir_2 \leq 230.5$; $mir_1 \leq 336.5$; $mir_2 > 199.5$; $\Delta_{21}(nir) > -83.5$ then 0, else
4	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 \leq 150.5$ then 0, else
5	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 \leq 348.5$; $\Delta_{21}(nir) \leq -52.5$; $red_1 \leq 158.5$ then 1, else
6	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 \leq 348.5$; $\Delta_{21}(nir) \leq -52.5$; $red_1 > 158.5$ then 0, else
7	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 \leq 348.5$; $\Delta_{21}(nir) > -52.5$ then 0, else
8	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 > 348.5$; $\Delta_{21}(nir) \leq -87.5$; $red_2 \leq 108.5$ then 0, else
9	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 > 348.5$; $\Delta_{21}(nir) \leq -87.5$; $red_2 > 108.5$ then 1, else
10	$Nir_2 \leq 230.5$; $mir_1 > 336.5$; $mir_2 > 150.5$; $mir_2 > 348.5$; $\Delta_{21}(nir) > -87.5$; then 0, else
11	$Nir_2 > 230.5$ then 0, else 0

To process the data the c-shell script gba_uoe_processor.csh is used (Annex J). The input variables to the script are as follows:

```
#######################################################
GBA2000 C shell: gba_uoe_processor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_uoe_processor.csh
<acronym>
<path to gba_uoe_br_ba_algorithm.pro>
<path to IDL executable binary>
<number of files to classify (e.g. 84 for the year 2000)>
```

where, acronym indicates the region being processed (e.g. BR). The number of files contained within a text file indicating the files to be processed (monthly composites) is also given at the command line. A complete description is given in Annex J.

A text file indicating the files that are to be processed should be copied or created in the working directory and given the name gba_uoe_month_list.txt. The file resembles the following (for the example of window BR):

```
BR_19991221_31_b2_all_minNIR
BR_19991221_31_b3_all_minNIR
BR_19991221_31_mir_all_minNIR
BR_20000101_10_b2_all_minNIR
BR_20000101_10_b3_all_minNIR
BR_20000101_10_mir_all_minNIR
BR_200001_basic_nir
…
BR_20001201_10_mir_all_minNIR
BR_200012_basic_nir
```

The format of the text file (in the example shown) informs the program that the composite at time t-1 comprises the month of December 1999 and the composite at time t comprises the month of January 2000. The seventh file in the list indicates the name of the output file that will be written to a new directory named ba_uoe. The c-shell script calls an IDL program, containing the burnt area algorithm. The program is called gba_uoe_br_ba_algorithm.pro and is described in Annex J.

11.4 UOE algorithm post-processing procedure

A number of post-processing procedures were applied to the monthly burnt area products for the BR window. All of the post-processing steps are undertaken using a c-shell script named gba_uoe_postprocessor.csh (Annex J). The input variables to the script are as follows:

```
#######################################################
GBA2000 C shell: gba_uoe_postprocessor.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_uoe_processor.csh
<acronym>
<region grow value (see ba_expand.c in ~tanseke/bin)>
```

where, acronym indicates the region being processed (e.g. BR) and the region grow value is the number corresponding to the number of burnt pixels that must surround a burnt pixel for all eight pixels in a 3x3 window surrounding the burnt pixel to all be designated as being burnt. A complete description is given in Annex J.

Before the post-processing is undertaken, several auxiliary files are needed. These include bi-monthly (a composite of months 1 and 2) sun shadow masks (located in directory ./sun_shadow_masks) and a greater than ten degree slope mask (located in directory ./dem_products). Bi-monthly sun shadow masks are used in this example because the sun shadowing can influence composites of both months. The method of deriving these products is described in Section 1.5. In this section, it is assumed that these products have already been derived for the region of interest.

The first step in the post-processing procedure is the masking out of those pixels potentially affected by a reduction in the solar radiance caused by topography. Also pixels with a calculated average slope value in excess of ten degrees are masked. The value of ten degrees was chosen after a test was made of the detected burnt areas with Landsat TM quicklooks. The second step was the summing of monthly products to test for the occurrence of pixels detected as being burnt twice in the year 2000. For the region of Brazil and surrounding countries (i.e. window BR) only a very small number of pixels (< 20) actually burnt twice during the year 2000 even in the northern part of the window. The main burning activity here started in January and February. A double detection mask was derived, that removed all of those pixels detected twice in the year 2000 and this was applied to each of the monthly products. The third step was to grow in size the large burnt area clumps observed in the monthly maps. This step was made because after comparisons of the burnt area maps with those derived from the original algorithm (composites analysed over 3 months) were made it was found that the well-defined burn scars were not as completely mapped with the modified algorithm. The burnt areas would not be as big or as complete with the modified algorithm. However, the burnt areas represented by single pixels complemented each other well with

both algorithms. A program was written that for each pixel determined as burnt, if those surrounding pixels in a 3x3 window exceeded a certain number (set by the user) then all those pixels in the 3x3 window would be indicated as being burnt. The program named ba_expand.c is described in Annex A. For the window BR, the value chosen was greater than four. This means that of the eight surrounding pixels, if five or more are indicated as being burnt then all eight would be indicated as being burnt in the output product. The value of four was selected after tests were made of selected burnt areas in the expanded product versus the original product.

The fourth step was to once again remove the burnt pixels characerised by large slope angles or possibly affected by sun shadow. This is applied again because it is possible that some areas previously masked out in step two, are now again indicated as being burnt by the processing in step 3 (burnt area region growing). Step five involves the derivation of unique monthly burnt area maps using two programs, unique_ba_maps_1.c and unique_ba_maps_2.c described in Chapter 13. The final step is to extract the unique region of interest from the buffered window. This is done using the c-shell script gba_snip_2_unique_win.csh also described in Chapter 13. The processing is now finished and the final, non-accumulative monthly burnt area maps have been produced.

12 GVM (Boschetti) Algorithm Module

The algorithm developed for Mexico and India has been developed by Luigi Boschetti while working in the GVM unit of the Joint Research Centre. It is an improvement of the NRI change detection algorithm, adopting the change detection strategy introduced by Roy *et al*. (2002). In addition, the variation of the signal due to the change in viewing geometry is modelled rather than minimising its effect through a sampling strategy, as done in the NRI algorithm described in Chapter 7 (Boschetti *et al*., 2002). The algorithm has been developed, and calibrated over two geographical regions, namely the central part of Mexico and India and the results from Mexico are used in the GBA2000 product.

12.1 GVM (Boschetti) module

The algorithm requires three main stages that are preformed independently:

1) Pre-processing for cloud and noise masking
2) Pre-processing for BRDF inversion
3) Burnt area detection

The burnt area algorithm exploits the fact that a vegetation fire drastically changes the spectral characteristics of the local environment and by looking at pre-burn and post-burn conditions these changes, that are greater than changes brought on by other factors, can be identified. The main features and assumptions of the approach taken are:

- To apply a change detection algorithm it is assumed that the changes in the spectral radiance due to the land cover change induced by a fire are significantly greater than the changes due to other factors (Ingram *et al*., 1981). The spectral channels corresponding to the near-infrared (B3) and middle-infrared (MIR) are used to detect the changes resulting from a fire event, following the results and recommendations of Trigg and Flasse (2000).
- The algorithm makes use of BRDF semi-empirical (kernel based) models, where the reflectance R is modelled as (Lucht *et al*., 2000):

$$R(\theta,\vartheta,\phi,\Lambda)=\sum_k f_k(\Lambda),K_k(\theta,\vartheta,\phi).$$

where:

$R(\theta,\vartheta,\phi,\Lambda)$ is the BRDF in waveband Λ

θ is the solar zenith angle

ϑ is the view zenith angle

ϕ is the view-sun relative azimuth angle

$f_k(\Lambda)$ is the BRDF kernel model parameter k in waveband Λ

$K_k(\theta,\vartheta,\phi)$ is the BRDF model kernel *k*

The advantage of this particular class of models is that, as the BRDF is a linear function of the k parameters and consequently the parameters can be retrieved by means of a least squares fit.

- The BRDF model adopted is the Roujean model (Roujean *et al*., 1992), which has been already successfully applied to SPOT-VEGETATION data (Duchemin *et al*., 2002a; Duchemin *et al.*, 2002b). It models *R* as:

$$R(\theta,\vartheta,\phi,\Lambda)=f_0(\Lambda)+f_1(\Lambda)k_1(\theta,\vartheta,\phi)+f_2(\Lambda)k_2(\theta,\vartheta,\phi).$$

$$k_1(\theta,\vartheta,\phi)=\frac{1}{2\pi}[(\pi-\phi)\cos\phi+\sin\phi]\tan\theta\tan\vartheta-\frac{1}{\pi}[\tan\theta+\tan\vartheta$$
$$+\sqrt{(\tan\theta)^2+(\tan\vartheta)^2-2}]$$

$$k_2(\theta,\vartheta,\phi)=\frac{4}{3\pi(\cos\theta+\cos\vartheta)}\left[\left(\frac{\pi}{2}-\xi\right)\cos\xi+\sin\xi\right]-\frac{1}{3}$$

where:

$\xi=\arccos(\cos\vartheta\cos\theta+\sin\theta\sin\vartheta\cos\phi)$ is the phase angle.

As the two k are deterministic functions of the angles, which come with the S1 products as ancillary data, the model is relatively easy to implement and the inversion has a low computational cost. The [f_0, f_1, f_2] parameters are retrieved for the B3 and MIR bands only. The inversion is performed daily using a moving window of m days.

- The change detection technique (Roy *et al*., 2002) is based on the comparison between the new (or current) image of day t and the image obtained applying the model in direct mode, using the viewing geometry of time t and the BRDF parameters derived from the inversion of the model with the observations available at time t-1.
- For the algorithm to work, it is very important to remove pixels contaminated by cloud, smoke and cloud shadow from the dataset before the BRDF inversion. The

contaminated data are masked out using the pre-processing module developed specifically for the GBA2000 project (see Chapter 3).

12.2 GVM (Boschetti) algorithm implementation

12.2.1 Inversion of the GVM (Boschetti) model

The linear model is inverted through a least squares fit, based on the last m days, provided that among these at least n (n≤m) are available (cloud/noise free). For the processing of the Mexican window of GBA 2000, m was set to 15 and n to 8.

Using the same notation as before, if R(t, Λ) is the reflectance observed for a single pixel at day t for the band Λ, it is possible to write the p (n≤p≤m) equations:

$$R(t_1, \Lambda) = f_0 + k_1(t_1) f_1 + k_2 (t_1) f_2$$

$$R(t_2, \Lambda) = f_0 + k_1(t_2) f_1 + k_2 (t_2) f_2$$

.....

$$R(t_p, \Lambda) = f_0 + k_1(t_p) f_1 + k_2 (t_p) f_2$$

Which can be written as a relation between vectors:

$$R_{p,1}=A_{p,3} f_{3,1}$$

As the vector R is the vector of the observations and the matrix A is a deterministic function of the ancillary data, the vector f can be estimated as:

$$\hat{f} = (A^T A)^{-1} A^T R$$

As the model is strictly pixel based, each pixel is processed independently and no ancillary data on vegetation cover is required.

The use of least squares fit to invert the model imply that there are no changes due to phenology, greening or drying of the vegetation, which can be detected at the time scale of the moving window. This means that the changes over m days, due to the above-mentioned causes, must be negligible. At the same time it must be noticed that to avoid inconsistent results, the parameter n sets the minimum number of observations for the regression to be performed. In addition, in case of persistent cloud cover, as in many regions of the globe, m and n should be set after an analysis of the number of consecutive cloud-free days that one might expect to have when processing the time series. Finally, the moving window

approaches, which have been adopted, have higher computational cost, but lower risk of no data in the final product, due to cloud cover.

The inversion module of the processing chain is split in two parts:

- Roujean_coefficients.pro computes the two kernels of Roujean's model, which are used both for the inversion and for the subsequent application of the model in direct mode. The inputs are the four angle bands and the output products are two matrices of coefficients, the same size as the input data and in floating point data format. A full description of this IDL program is available in Annex K.
- Bidirectional_correction.pro performs the actual inversion of the model. The program requires as input, the time series of the spectral bands, whose inversion is required, the two bands of Roujean's coefficients and the angles bands. The program saves as output the three functions f_0 f_1 f_2 (floating point data type), one for each day and for each input spectral band. A full description of this IDL program is available in Annex K.

12.2.2 GVM (Boschetti) burnt area detection strategy

Burned areas are detected through a change detection procedure based on the analysis of a difference image. Instead of applying one of the traditional methods for the production of the difference image (for a review, see Coppin *et al*., 2002.), it is obtained with the method introduced by Roy *et al*. (2002), which takes explicitly into account the variation of the reflectance due to the variation of the viewing and illumination geometry.

The difference image D(t), representative of the change between time t-1 and time t, is obtained as:

$$\mathbf{D}(t) = \mathbf{R}(t|t\text{-}1) - \mathbf{R}(t)$$

where, R(t) is the reflectance observed at time t and R(t|t-1)=[1, k1(t), k2(t)] $\hat{f}(t-1)$, where R(t|t-1) is obtained applying the model in direct mode using the kernels with the viewing condition of day t and the parameters f estimated at time t-1. In other words, it is the reflectance expected at time t on the basis of the information available *a priori*, i.e. the what we would expect to observe if nothing had changed between time t-1 and time t. As a consequence, the difference image will highlight any change occurred between t-1 and t, without being affected by the variation of reflectance due to the different viewing conditions.

The analysis of the difference image, for the actual burned area detection, is performed using the same approach adopted in the NRI algorithm for South Africa, described in chapter 7 of this report. This approach requires the use of deterministic thresholds, which are determined making use of the *a priori* knowledge on the spectral signature of burned areas (Stroppiana *et al*., 2002; Trigg and Flasse, 2001). This approach requires considerably fewer computational resources than more advanced methods (Bruzzone and Prieto, 2000) and, though occasionally less accurate, is more reliable for the unsupervised processing of a long time series of data, like those in the GBA2000 dataset.

The burned area detection is as follows:

- A reference image is made from the time series of f(t): it is initiated at the beginning of the processing, and it is updated at every step with those pixels that are available (i.e. where enough observations were available to invert the model) and that have not been marked as potential burned areas. The difference image has six separate layers, as each band (in this study only B3 and MIR bands have been used for the burned area detection) requires the three f coefficients.
- The difference image (which is made by only two layers, one for each band) is computed pixel by pixel, as described before.
- A pixel is marked as potentially burned if its value in the difference image is greater than the fixed thresholds mir_min and b3_min (i.e. a decrease of the reflectance has occurred in both bands).
- A pixel is marked as burned if it has been already marked as potentially burned in the previous day and if the difference image is still greater than mir_min and b3_min, i.e. if the decreasing of the signal is persisting. The date, when the pixel has been marked as potentially burned is recorded in the output burned area matrix.
- The difference image is updated, taking from the new f(t) the pixels with non-nil value (i.e. the pixels where n observations were available, and where the inversion had been performed) and not marked as potentially burned.

The burned area detection is performed by the IDL program ba_detection.pro (described in Annex K) whose inputs are a time series of the relevant bands, a time series of f coefficients for the same bands and a time series of k_1 and k_2 kernels. The output of the program is a cumulative burned area map (of integer data type) where pixels assigned a value of zero are

not burned, otherwise burned pixels are given a value corresponding to the date (month year) of first detection.

12.3 Further developments in the GVM (Boschetti) algorithm

The algorithm has been implemented using a multi-stage structure, where the output of each stage is saved to disk and used as input of the following stage. This means that improvements and modifications on one single module do not require any modification in the others, so long as the output / input formats are not modified. Further investigation will be devoted to:

- sensitivity to cloud masking
- detection of outlyers (anoumalous data / clouds not detected etc.) during the inversion
- different kernels for the inversion
- sensitivity analysis of the final burned area detection.

13 Creating the global burnt area product

This chapter describes the steps taken to create the global burnt area product. Such steps include the extraction of the unique (non-buffered) region of interest, the detection and masking of burnt area pixels detected in consecutive months (a product of some of the algorithms), the mosaicking of the windows into larger or global burnt area maps and the creation of ARC-INFO GRID files that are used to import into ARC-VIEW or a similar GIS. The usage of the regional algorithms to process regions outside of their development is also presented in this chapter.

13.1 Extracting the unique region of interest from the buffered window

Due to the problems of cloud shadow, it was necessary to extract regions of interest with a buffer zone of twenty pixels. This buffer zone was then removed to derive the unique burnt area maps. When processing data with the IFI algorithm this task is performed automatically after the pre-processing stages. For the other algorithms and in particular those that require cloud shadow screening (e.g. UTL and GVM) this task is undertaken at the end of the processing. A c-shell script is available to carry out this task called gba_snip_2_unique_win.csh (see Annex L). The input parameters to this program are:

```
#######################################################
GBA2000 C shell: gba_snip_2_unique_win.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_snip_2_unique_win.csh
<acronym (e.g. SA, AU, BR)>
<algorithm (e.g. utl, gvm, uoe)>
```

where, the acronym of the monthly burnt area maps and the algorithm that has been used to process the data are given. The program has been created to interpret the different naming conventions used for the different algorithms.

13.2 Checking for double burnt area detections in monthly products

For some algorithms, the presence of the same pixel indicated as being burnt in consecutive months is something that needs to be corrected for in the burnt area products. With regard to the original objective, the monthly burnt area maps are non-accumulative. However, in some cases the pixels under observation can be burnt twice during the year 2000. This is the case in

sub-Saharan Africa north of the equator and in Asia. Pixels could be indicated as being burnt at the end of the 1999-2000 fire season and again at the beginning of the 2000-2001 fire season. It is important that the monthly products reflect this occurrence. For the annual product, this double burn cannot be indicated so this product simply reflects the fact that a pixel burnt during the year 2000, regardless of the timing or the frequency. To remove double detections from consecutive months during the same fire season two programs are available to yield non-accumulative monthly burnt area products. The first program called unique_ba_maps_1.c is applied to the second map in the series. Obviously, the first map is the reference image so that all those pixels shown as being burnt in the first image cannot be burnt in the second image. The output of this first program is a unique burnt area map for the second map and a mask indicating those pixels that have burnt during months one and two. The second program called unique_ba_maps_2.c takes consecutive monthly products and the mask of those areas previously burnt and applies this to the map under observation. The output of this program is a unique monthly product and an updated version of the mask containing all those pixels identified as being previously burnt. The programs are presented in full in Annex L and an example shown in Annex H. Examples of the input commands to these programs are:

unique_ba_maps_1 CA_20001101_31_utl_ba ca_10 ca_11 mask_10_11 $pixels $lines
unique_ba_maps_2 CA_20001201_31_utl_ba ca_mask_10_11 ca_12 mask_10_11_12 $pixels $lines

13.3 Mosaicking of processed windows

It is necessary for the GBA2000 project to create regional mosaics from the processing windows defined in Chapter 1. A program is available to mosaic two images together. In this program those of the image being created replace the values in the original image. Both files must already exist. The large window can be created using the generic c program make_raster (Annex A). In addition, two text files must exist corresponding to each window. The program uses information from these text files to calculate the offset and location of the two images. Once the monthly products are mosaicked the annual, burnt area map can be created by using the generic c program add_files.c (Annex A). The program that performs the mosaicking is called gba_ba_mosaic.c (see Annex L) and an example of the input command of this program to mosaic the tropical African window (TA) into the large sub-Saharan (d) window is as follows:

gba_ba_mosaic
d_05 $pixels_d $lines_d $global_x_offset_d $global_y_offset_d
ta_05 $pixels_ta $lines_ta $global_x_offset_ta $global_y_offset_ta
d_05_new

13.4 Generating ARC-INFO GRID products

ARC-INFO GRID files can be generated automatically by using a workstation with access to an ARC license. GRID format files are useful as they can be transferred across different systems and over networks without concern about loss of information. They can be easily imported into ARC-VIEW and other GIS software packages. To create GRID files, a header file needs to be available that is recognised by the ARCINFO software. This text file provides important information on the geometry, size (in bytes), byte and band order and projection information. For the production of GRID files this projection information refers to the centre of the upper left pixel (for byte or integer format data). Note that a different program is used for importing floating point data. An example of an ARCINFO header file is presented below for the sub-Saharan window:

nrows 6000
ncols 8200
nbands 1
nbits 8
layout bsq
byteorder m
ulxmap -17.946429
ulymap 18.303572
xdim 0.0089285714
ydim 0.0089285714

where, nrows and ncols refer to the geometry of the file, nbands if the number of bands, nbits is the number of bits per pixel of the image, layout is the band order, in this case band sequential (BSQ), byteorder refers to the structure of the data, in this case the m stands for motorolo (or IEEE) as opposed to INTEL format, the ulxmap and ulymap refers to the map coordinates, in this case the centre of the upper-left pixel and xdim and ydim refer to the pixel spacing.

A c-shell script is available to automatically create GRID format products of the monthly and annual burnt area products. The script is called gba_create_grids.csh and uses ARC-INFO commands (see Annex L). An example of the input commands to this script are:

```
#######################################################
GBA2000 C shell: gba_create_grids.csh
Copyright JRC, 2001
Contact person: kevin.tansey@jrc.it
#######################################################
Syntax: gba_create_grids.csh
<acronym of info text file (e.g. af, a, b, c, etc.)>
<file naming convention (e.g. utl or enter 0)>
<path to ARCINFO executable>
```

where, acronym refers to the region of interest (e.g. d corresponding to the sub-Saharan Africa region), the file naming convention refers to the algorithm used to create the product (if this does not exist enter a zero value here) and the path refers to the full directory path of the ARC-INFO executable. Please note that lower case letters are required for all filenames for this procedure to work.

The program automatically creates the header file required and renames the files with the correct .bsq extension. The program looks for each monthly product and finally the annual product for the year 2000. The resultant GRID files are written to their own specific directory within a newly created directory called grids. These files can then be imported into ARC-VIEW or another GIS software. Once imported these GRID files can be re-projected to an equal-area projection or resampled to a coarser resolution.

13.5 Producing the GBA2000 burnt area map

In Chapter 1, the division of the globe into processing windows was described. This was to ensure that the algorithms could be successfully applied without causing excessive demands on memory and disk space. The selection of the regional algorithm, to process windows not covered in the algorithm development phase, was made through observation of the land cover properties, the seasonality of fire occurrence and by including factors such as phenology and snow cover. Table 4 shows where the algorithms were used to produce the GBA2000 product. The months processed related to the fire season and the acronym of the final burn area product that the sub-window are included. Firstly though, a reminder is given of the sub-windows processed by the algorithms for the GBA2000 product (Figure 37).

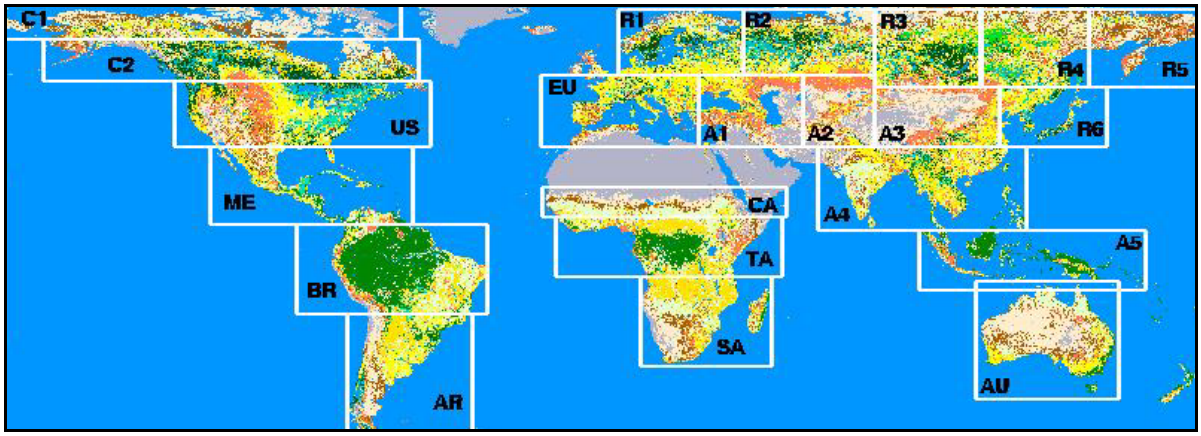


Figure 37: Division of the global dataset into sub-windows, imposed because of hardware and software processing limitations. An acronym is given for each sub-window.

From these sub-windows, regional windows were constructed that were used for data dissemination. A global product was also created. The six regional windows constructed from these sub-windows are shown in Figure 38.

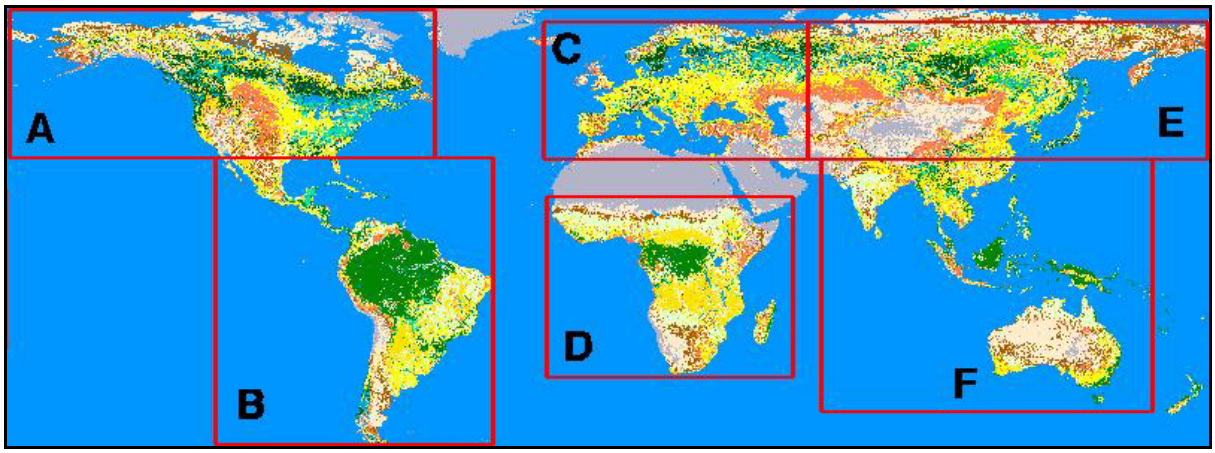


Figure 38: Division of the global dataset into regions. An acronym is given for each region.

These can be described as covering northern America (A); central and southern America (B); Northern Africa, Europe and the Middle East (C); sub-Saharan Africa (D); northern Asia (E); southern Asia and Australia (F). Some regions of the globe are not processed, namely Greenland, sections of Great Britain and Ireland, New Zealand, the section of Russia located in the Western hemisphere and the Saharan region of Africa and the Arabian Peninsula. It was believed that burning activity is so small or the vegetation cover is so low in these regions that they did not require any attention.

Table 4: The selection of the regional algorithm as applied to each of the sub-windows shown in Figure 37.

Sub-Window (see Figure 37)	Months Processed	Algorithm(s) Used	Regional Window (see Figure 38)
C1	21/04 – 30/09	CCRS	A
C2	21/04 – 30/09	CCRS	A
US	21/04 – 10/10	CCRS (forests); UTL - Africa (non-forests)	A
ME	01/10 – 31/05	IFI (all summer algorithm) + GVM (Boschetti)	B
BR	01/01 – 31/12	UOE/UTL	B
AR	01/01 – 31/12	IFI (autumn algorithm used 04 – 09/2000)	B
EU	01/04 - 31/10	IFI (no sieve used, 3x3 expansion in forest only) UTL – Europe to 48^{o} N (06 – 09/2000)	C
R1	01/04 - 31/10	IFI (3x3 expansion in forest only)	C
R2	01/04 - 31/10	IFI (3x3 expansion in forest only)	C/D
R3	01/04 - 31/10	IFI (3x3 expansion in forest only)	D
R4	01/04 - 31/10	IFI (3x3 expansion in forest only)	D
R5	01/04 - 31/10	IFI (3x3 expansion in forest only)	D
R6	01/04 - 31/10	IFI (3x3 expansion in forest only) IFI + UTL – Asia (05, 07, 08/2000)	D
A1	01/04 - 31/10	IFI (3x3 expansion in forest only) IFI + UTL – Africa1 (04 – 09/2000) (UTL mask applied in 09/2000)	C
A2	01/04 - 31/10	IFI (3x3 expansion in forest only) (post-proc. factor of 0.2 used in 09, 10/2000)	D
A3	01/04 - 31/10	IFI (3x3 expansion in forest only) IFI + UTL – Asia (05, 06/2000) (mask applied to UTL in 05/2000) (post-proc. factor of 0.5 used in 09/2000)	D
A4	01/10 – 31/05	IFI (summer algorithm used throughout)	F
A5	01/01 – 31/12	IFI	F
AU	01/01 – 31/12	GVM (Stroppiana) (No tropical savanna product for 02, 03/2000)	F
CA	01/01 – 05/31 01/10 – 31/12	UTL – Africa2	D
TA	01/01 – 31/12	UTL – Africa2	D
SA	01/05 – 31/12	UTL – Africa2	D

The following figures illustrate the spatial and temporal distributions of the choice of algorithm implemented in the GBA2000 project. Figure 39 shows which algorithm was, or algorithms were, applied to each geographical regions of the globe. Figure 40 shows those months of the year 2000 when the algorithms were applied.

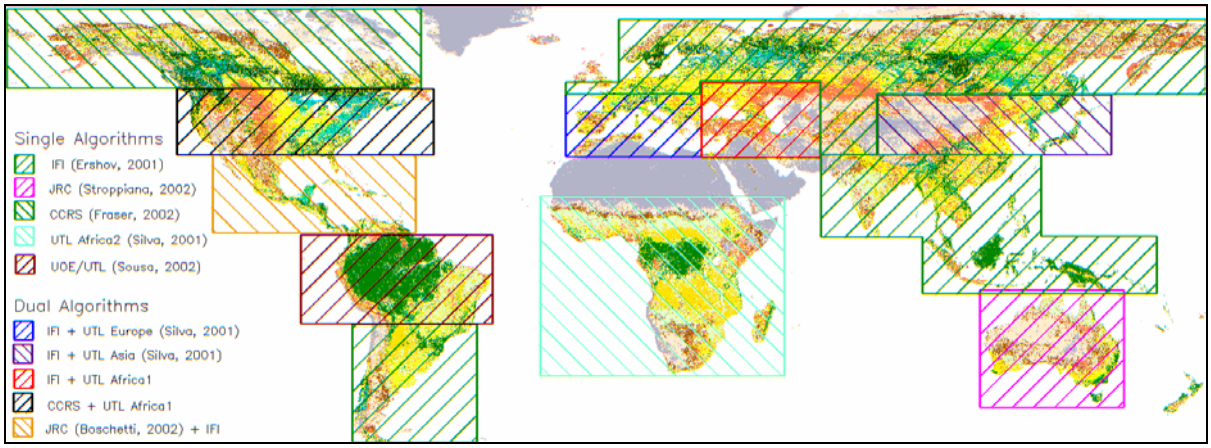


Figure 39: Where the GBA2000 algorithms are applied. Note that UTL Africa1 refers to the original UTL algorithm developed for GBA2000 (using linear discriminant analysis). Africa2 refers to the second algorithm developed by UTL (post-beta version) (using CART).

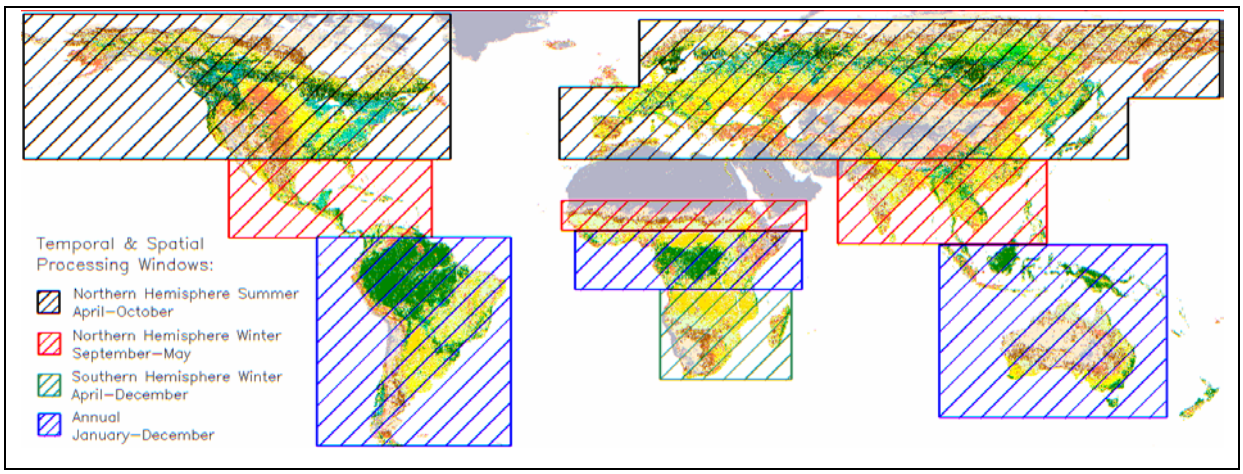


Figure 40: The months in the year when the algorithms are applied.

References

Arino, O., Simon, M., Piccolini, I., and Rosaz, M., 2001, The ERS-2 ATSR-2 World Fire Atlas and the ERS-2 ATSR-2 World Burnt Surface Atlas projects. Paper presented at *The 8th ISPRS conference on Physical Measurement and Signatures in Remote Sensing*, Aussois, January 8-12, 2001.

Barbosa P.M., Stroppiana D., and Grégoire, J-M., 1999, An assessment of vegetation fire in Africa (1981-1991): Burned areas, burned biomass, and atmospheric emissions. *Global Geochemical Cycles*, Vol. 13, pp. 933-950.

Belward, A. S. (ed.), 1996, The IGBP-DIS global 1 km land cover dataset (DISCover) proposal and implementation plans. *IGBP-DIS Working Paper No. 13*, Toulouse, France, p. 61.

Belward, A.S., Estes, J.E., and Kline, K.D., 1999, The IGBP-DIS 1 km land cover dataset DISCover: A project overview. *Photogrammetric Engineering and Remote Sensing*, Vol. 65, pp. 1013-1020.

Binaghi E., Brivio, P.A., Ghezzi, P., Rampini, A., and Zilioli, E., 1999, Investigating the behaviour of neural and fuzzy-statistical classifiers in sub-pixel land cover estimations. *Canadian Journal of Remote Sensing*, Vol. 25, pp. 171-188.

Boschetti L., Flasse, S., Jacques de Dixmude, A., and Trigg, S., 2002, A multitemporal change-detection algorithm for the monitoring of burnt areas with SPOT-Vegetation data. In: *Analysis of Multi-temporal Remote Sensing Images*, Bruzzone, L. and Smith, P. (Eds.), Singapore: World Scientific Publishing, pp.75-82.

Breiman, L., Friedman, J.H., Olshen, R.A., and Stone, C.J., 1984, *Classification and Regression Trees*. Belmont, CA: Wadsworth Int., 1984.

Brems E., Lissens, G., and Veroustraete, F., 2000, MC-FUME: a new method for compositing individual reflective channels. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 38, pp. 553-569.

Brivio, P.A., Binaghi, E., Gallo, I., Giradino, C., and Maggi, M., 2001a, Burnt area mapping: Central Africa. *End of Contract Report (Contract No.: 16180-2000-05 F1 EI ISP IT)*, CNR-IREA, Milano.

Brivio P.A., Maggi, M., Binaghi, E., Gallo, I., and Grégoire J-M., 2001b, Exploiting spatial and temporal information for extracting burned areas from time series of SPOT-VGT data. In: *Analysis of Multi-temporal Remote Sensing Images*, Bruzzone L. and Smits P. (eds.), World Scientific, Singapore, pp. 132-139.

Brivio P.A., Binaghi, E., Gallo, I., and Maggi, M., (in press), Contextual multi-temporal classification of burned areas in coarse resolution imagery. In: *Geospatial Pattern Recognition*, Binaghi E., Brivio, P.A., and Serpico S.B. (eds), Transworld Research Network, Trivandrum, India.

Brivio P.A., Maggi, M., Binaghi, E., and Gallo, I., (submitted), Mapping burned surfaces in Sub-Saharan Africa based on multi-temporal neural classification. *International Journal of Remote Sensing*.

Bruzzone, L. and Prieto, D.F., 2000, Automatic analysis of the difference image for unsupervised change detection. *IEEE Transaction on Geoscience and Remote Sensing,* Vol. 38, pp. 1171-1182.

Cabral, A., de Vasconcelos, M.J.P., Pereira, J.M.C., Bartholomé, E., and Mayaux, P., (accepted), Multitemporal compositing approaches for SPOT-4 VEGETATION data. *International Journal of Remote Sensing*.

Cihlar J., Manak, D., and D'Iorio, M., 1994, Evaluation of compositing algorithms for AVHRR data over land. *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 32, pp. 427-437.

Colby, J. D., 1991, Topographic normalization in rugged terrain. *Photogrammetric Engineering and Remote Sensing*, Vol. 57, pp. 531-537

Coppin, P., Lambin, E., Jonckheere, I., and Musyb, 2001, Digital change detection in natural ecosystem monitoring: a review. In: *Analysis of Multi-temporal Remote Sensing Images*, Bruzzone, L. and Smith, P. (Eds.), Singapore: World Scientific Publishing, pp. 3-36

DeFries, R., Hansen, M., Townshend, J. R. G., and Sohlberg, R., 1998, Global land cover classifications at 8 km spatial resolution: The use of training data derived from Landsat imagery in decision tree classifiers. *International Journal of Remote Sensing*, Vol. 19, pp. 3141-3168.

De Wasseige, C., Vancutsem, C., and Defourny, P., 2000, Sensitivity analysis of compositing strategies: Modelling and experimental investigations. In: *Proceedings of VEGETATION 2000, 2 years of operation to prepare the future*, International Users Committee – Vegetation Program, Toulouse, France, pp. 267-274.

Duchemin, B., Maisongrande, P., Dedieu, G., Leroy, M., Roujean, J.L., Bicheron, P., Hautecoeur, O., and Lacaze, R., 2000, A ten days compositing method accounting for bidirectional effects. In: *Proceedings of VEGETATION 2000, 2 years of operation to prepare the future*, International Users Committee – Vegetation Program, Toulouse, France, pp. 313-318.

Duchemin, B. and Maisongrande, P., 2002a, Normalisation of directional effects in 10-day global syntheses derived from VEGETATION/SPOT: 1. Investigation of concepts based on simulation. *Remote sensing of the Environment*, Vol. 81, pp. 90-100

Duchemin, B., Berthelot, B., Dedieu, G., and Maisongrande, P., 2002, Normalisation of directional effects in 10-day global syntheses derived from VEGETATION/SPOT: II. Validation of an operational method on actual data sets. *Remote sensing of the Environment*, Vol. 81, pp. 101-113

Dwyer E., Grégoire, J-M., and Malingreau, J.P., 1998, A global analysis of vegetation fires: Spatial and temporal dynamics. *AMBIO*, Vol. 27, pp. 175-181.

Dwyer E., Pereira, J.M.C., Grégoire, J-M., and DaCamara, C.C., 1999, Characterization of the spatio-temporal patterns of global fire activity using satellite imagery for the period April 1992 to March 1993. *Journal of Biogeography*, Vol. 27, pp. 57-69.

Ershov, D.V. and Novik, V.P., 2001, Features of burnt area mapping in forest of Siberia using SPOT S1-VGT data. In: *Proceedings of the GOFC Fire Satellite Product Validation Workshop*, Lisbon, July 9-11 2001.

Fraser, R.H., Fernandes, R., and Latifovic, R., (submitted), Multi-temporal mapping of burned forest over Canada using satellite-based change metrics. *Geocarto International*.

Gao, B.-C., 1996, NDWI - A normalized difference water index for remote sensing of vegetation liquid water from space. *Remote Sensing of Environment*, Vol. 58, pp. 257-266.

Hansen, M., DeFries, R., Townshend, J. R. G., and Sohlberg, R., 2000, Global land cover classification at 1 km resolution using a decision tree classifier. *International Journal of Remote Sensing*, Vol. 21, pp. 1331-1365.

Hastings, D. A. and Dunbar, P. K., 1998, Development and assessment of the Global Land One-km Base Elevation Digital Elevation Model (GLOBE). *International Society of Photogrammetry and Remote Sensing Archives*, Vol. 32, pp. 218-221.

Hu, B., Lucht, W., Strahler, A.H., Scaaf, C.B., and Smith, M., 2000, Surface albedos and angle corrected NDVI from AVHRR observations of South America. *Remote Sensing of Environment*, Vol. 71, pp. 119-132.

Ingram, K., Knapp, E., and Robinson, J.W., 1981, Change detection procedure development for improved urbanized area delineation. *Technical memeorandum CSC/TM-81/6087*, Computer Sciences Corporation, Silver Springs, Maryland, U.S.A.

Johnson, R.A. and Wichern, D.W., 1988, *Applied Multivariate Statistical Analysis (Second Edition)*. Prentice Hall: New Jersey.

Kempeneers P., Lissens, G., Fierens, F., and Van Rensbergen, J., 2000, Development of a cloud, snow and cloud shadow mask for VEGETATION imagery. In: *Proceedings of VEGETATION 2000, 2 years of operation to prepare the future*, International Users Committee – Vegetation Program, Toulouse, France, pp. 302-306.

Lissens, G., Veroustraete, F., and van Rensbergen, J., 2000, MC-FUME: A new mothod for compositing individual reflective channels. In *Proceedings of VEGETATION 2000, 2 years of operation to prepare the future*, International Users Committee – Vegetation Program, Toulouse, France, pp. 281-285.

Lucht, W, Schaaf, C.B., and Strahler, A.H., 2000, An algorithm for the retrieval of albedo from space using semi-empirical BRDF models. *IEEE Transaction on Geoscience and Remote Sensing*, Vol. 38, pp. 977-998.

Olson, J.S., 1994a, Global ecosystem framework definitions. *USGS EROS Data Centre Internal Report*, Sioux Fall, SD, p. 37.

Olson, J.S., 1994b, Global ecosystem framework translation strategy. *USGS EROS Data Centre Internal Report*, Sioux Fall, SD, p. 39.

Paola J. D. and Schowengerdt, R.A., 1995, A review and analysis of back-propagation neural networks for classification of remotely-sensed multi-spectral imagery. *International Journal of Remote Sensing*, Vol. 16, pp. 3033-3058.

Pereira, J.M.C., Vasconcelos, M.J.P., and Sousa, A.M., 2000, A ruled-based system for burned area mapping in temperate and tropical regions using NOAA/AVHRR imagery. In *Biomass burning and its inter-relationships with the climate system*, Innes, J.L., Beniston, M. and Verstraete, M.M. (eds.), The Netherlands: Kluwer Academic Publishers, 2000.

Rahman, H. and Dedieu, G., 1994, SMAC: a simplified method for the atmospheric correction of satellite measurements in the solar spectrum. *International Journal of Remote Sensing*, Vol. 15, pp. 123-143.

Roujean, J.L., Leroy, M., and Deschamps, P.Y., 1992, A bi-directional reflectance model of the Earth's surface for the correction of remotely sensed data. *Journal of Geophysical Research*, Vol. 97(D18), pp. 20455-20468.

Roy, D., Lewis, P.E., Justice, C.O., 2002, Burned area mapping using multi-temporal moderate spatial resolutio data – a bi-directional reflectance model-based expectation approach. *Remote Sensing of the Environment*, pp. 263-286.

Sellers, P.J., Mintz, Y., Sud, Y.C., and Dalcher, A., 1986, A simple biosphere model (SiB) for use within general circulation models. *Journal of Atmospheric Science*, Vol. 43, pp. 505-531.

Sellers, P.J., Randall, D.A., Collatz, G.J., Berry, J.A., Field, C.B., Dazlich, D.A., Zhang, C., Collelo, G.D., and Bounoua, L., 1996, A revised land surface parametrization (SiB2) for atmospheric GCM's – Part I – model formulation. *Journal of Climate*, Vol. 9, pp. 676-705.

Shepherd, J.D. and Dymond, J.R., 2000, BRDF correction of vegetation in AVHRR imagery. *Remote Sensing of Environment*, Vol. 74, pp. 397-408.

Silva, J.M.N., Pereira, J.M.C., Cabral, A.I., Sá, A.C.L., Vasconcelos, M.J.P., Mota, B., and Grégoire, J.-M., 2002a, The area burned in southern africa during the 2000 dry season. *Journal of Geophysical Research*, SAFARI 2000 Special Issue.

Silva, J.M.N., Sousa, A.M.O., Pereira, J.M.C., Tansey, K., and Grégoire, J.-M., 2002b, A contribution for a global burned area map. In: *Forest Fire Research & Wildland Fire Safety,* Viegas, D. X. (ed.), Millpress, Rotterdam.

Stocks, B.J., 1991, The extent and impact of forest fires in northern circumpolar countries. In: *Global Biomass Burning: Atmospheric, Climate, and Biospheric Implications*, Levine, J.S. (ed.), MIT Press, Cambridge, MA.

Stroppiana D. and Grégoire, J-M., 2001, Using temporal change of the land cover spectral signal to improve burnt area mapping. In: *Analysis of Multi-temporal Remote Sensing Images*, Bruzzone, L. and Smith, P. (Eds.), Singapore: World Scientific Publishing, pp. 209-216.

Stroppiana, D., Pinnock, S., Pereira, J.M.C., and Grégoire, J.M., 2002, Radiometric analysis of Spot Vegetation images for burnt area detection in northern Australia. *Remote Sensing of Environment*, Vol. 82, pp. 21-37

Stroppiana D., Tansey, K., Grégoire, J-M., and Pereira, J.M.C., (accepted), An algorithm for mapping burnt areas in Australia using SPOT-VEGETATION data. *IEEE Transactions on Geoscience and Remote Sensing*.

Saunders, R.W., 1990, The determination of broad band surface albedo from AVHRR visible and near-infrared radiances. *International. Journal of Remote Sensing*, Vol. 11, pp. 59-67.

Trigg, S. and Flasse, S., 2000, An evaluation of different bi-spectral spaces for discriminating burned shrub-savanna. *International Journal of Remote Sensing*, Vol. 21, pp. 3161-3168.

Trigg, S. and Flasse, S., 2001, An evaluation of the bi-spectral spaces for discriminating burned shrub – savannah. *International Journal of Remote Sensing,* Vol. 22, pp. 2641-2647.

Wanner, W., Li, X., and Strahler, A.H., 1995, On the derivation of kernels for kernel-driven models of bidirectional reflectance. *Journal of Geophysical Research*, Vol. 100, pp. 21077-21090.