



fire
cci

ESA Climate Change Initiative – Fire_cci

D3.2 System Verification Report (SVR)

Project Name	ECV Fire Disturbance: Fire_cci Phase 2
Contract Nº	4000115006/15/I-NB
Issue Date	11/01/2019
Version	2.4
Author	Thomas Storm
Document Ref.	Fire_cci_D3.2_SVR_v2.4
Document type	Public

To be cited as: T. Storm, M.L. Pettinari, G. Otón, J. Lizundia-Loiola (2019) ESA CCI ECV Fire Disturbance: D3.2. Software Verification Report, version 2.4. Available from: <http://www.esa-fire-cci.org/documents>

 <div>fire cci</div>	<div>Fire_cci</div> <div>System Verification Report</div>	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4			
		Issue	2.4	Date	11/01/2019	
					Page	2

Project Partners

Prime Contractor/ Scientific Lead & Project Management	UAH – University of Alcala (Spain)
Earth Observation Team	UAH – University of Alcala (Spain)
	EHU – University of the Basque Country (Spain)
	UL – University of Leicester (United Kingdom)
	UCL – University College London (United Kingdom)
	ISA – School of Agriculture, University of Lisbon (Portugal)
System Engineering	BC – Brockmann Consult (Germany)
Climate Research Group	MPIC – Max Planck Institute for Chemistry (Germany)
	IRD - Research Institute for Development (France)
	LSCE - Climate and Environmental Sciences Laboratory (France)
	VUA - Stichting VU-VUmc (Netherlands)



Distribution

Affiliation	Name	Address	Copies
ESA	Stephen Plummer (ESA)	stephen.plummer@esa.int	electronic copy
Project Team	Emilio Chuvieco, (UAH)	emilio.chuvieco@uah.es	electronic copy
	M. Lucrecia Pettinari (UAH)	mlucrecia.pettinari@uah.es	
	Joshua Lizundia (UAH)	joshua.lizundia@uah.es	
	Aitor Bastarrika (EHU)	aitor.bastarrika@ehu.es	
	Ekhi Roteta (EHU)	ekhi.roteta@gmail.com	
	Kevin Tansey (UL)	kjt7@leicester.ac.uk	
	Marc Padilla Parellada (UL)	mp489@leicester.ac.uk	
	James Wheeler (UL)	jemw3@leicester.ac.uk	
	Philip Lewis (UCL)	ucfalew@ucl.ac.uk	
	José Gómez Dans (UCL)	j.gomez-dans@ucl.ac.uk	
	James Brennan (UCL)	james.brennan11@ucl.ac.uk	
	Jose Miguel Pereira (ISA)	jmocpereira@gmail.com	
	Duarte Oom (ISA)	duarte.oom@gmail.com	
	Manuel Campagnolo (ISA)	mlc@isa.ulisboa.pt	
	Thomas Storm (BC)	thomas.storm@brockmann-consult.de	
	Martin Böttcher (BC)	martin.boettcher@brockmann-consult.de	
	Johannes Kaiser (MPIC)	j.kaiser@mpic.de	
	Angelika Heil (MPIC)	a.heil@mpic.de	
	Florent Mouillot (IRD)	florent.mouillot@cefe.cnrs.fr	
	Philippe Ciais (LSCE)	philippe.ciais@lsce.ipsl.fr	
	Patricia Cadule (LSCE)	patricia.cadule@lsce.ipsl.fr	
	Chao Yue (LSCE)	chaoyuejoy@gmail.com	
	Pierre Laurent (LSCE)	pierre.laurent@lsce.ipsl.fr	
	Guido van der Werf (VUA)	guido.vander.werf@vu.nl	
	Ioannis Bistinas (VUA)	i.bistinas@vu.nl	

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		3

Summary

This document is the version 2.4 of the System Verification Report for the Phase 2 of the Fire_cci project. It documents the system verification activities of the ESA Climate Change Initiative (CCI) Fire project for its elements used in Phase 2.

The document defines the different verification approaches considered, and describes the approaches instrumented in every step of the BA product generation.

	Affiliation/Function	Name	Date
Prepared	System Engineer, BC Project Manager, UAH Researcher, UAH Researcher, UAH	Thomas Storm M. Lucrecia Pettinari Gonzalo Otón Joshua Lizundia-Loiola	10/01/2019
Reviewed	Scientific Leader, UAH Project Manager, UAH	Emilio Chuvieco M. Lucrecia Pettinari	11/01/2019
Authorized	Scientific Leader, UAH	Emilio Chuvieco	11/01/2019
Accepted	Technical Officer, ESA	Stephen Plummer	

This document is not signed. It is provided as an electronic copy.

Document Status Sheet

Issue	Date	Details
1.0	06/09/2016	First issue of the document
1.1	06/10/2016	Addressing ESA comments according to CCI_FIRE_EOPS_MM_16_0109.pdf
2.0	27/03/2017	Updated according to SFD processing
2.1	12/06/2017	Addressing ESA comments according to CCI_FIRE_EOPS_MM_17_0041.pdf
2.2	30/04/2018	Updated according to MODIS processing
2.3	02/11/2018	Updated according to AVHRR-LTDR processing and addressing comments of CCI_FIRE_EOPS_MM_18_0142.pdf
2.4	11/01/2019	Addressing ESA comments according to ESA-CCI-EOPS-FIRE-MEM-18-0200.pdf

Document Change Record

Issue	Date	Request	Location	Details
1.1	06/10/2016	ESA	Naming convention	Reference to the year of the project added to the name of the document.
			Executive Summary	Deleted. All section numbers were change accordingly.
			Sections 3.2, 3.3, 3.4	Minor changes in the text.
2.0	27/03/2017	UAH-BC	All document	Inclusion of verification steps of the Small Fire Database
2.1	12/06/2017	ESA	Sections 1.2 and 1.3	Subdivided previous Section 1.2 into two different sections.
			Sections 1.1, 2.1	Small changes in the text
			Section 3	New Section 3 added to better organize the information.
			Section 4.1.2	Added reference to Sen2Cor code.
			Section 4.2.1	Added information on the purpose of each test described in those sections.
			Sections 4.4.1, 4.4.2	Extended the explanation of the visual inspection.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		4

Issue	Date	Request	Location	Details
2.2	30/04/2018	BC	All document Section 1.4 Sections 2.3, 3.5, 4, 6.2.3, 6.3.3, 6.4.3, 7.3.3, 7.4.3 Section 7.2	Inclusion of verification steps of MODIS processing Document structure section deleted New sections added Section and sub-sections reorganized and updated.
2.3	02/11/2018	BC	All document Section 1.1, 6.1 Sections 2.4, 7.2.5, 7.3.4, 7.4.4 Sections 7.3.2, 7.3.3, 7.4	Inclusion of verification steps of MODIS and AVHRR processing, use of acronyms of BA products Small changes in the text Sections added Text updated
2.4	11/01/2019	UAH ESA UAH UAH, ESA UAH ESA ESA ESA	Sections 1.3, 2 Section 2.4, 3, 4.1.2, 6.1 Section 3.5 Sections 4.2, 6.2.3 Sections 6.3.4, 6.4.4 Sections 6.4.1, 6.4.3, 7.4.1, 7.4.2 Section 7.4.3 Annex 2	New references added Text updated to reference to the processing of the FireCCILT10 product. Section deleted Sections updated New sections added Clarification added Figure 7.19 updated and 7.21 added New annex added

Table of Contents

1	Introduction	8
1.1	Purpose of the document.....	8
1.2	Applicable Documents.....	8
1.3	Reference Documents	8
2	Systems under test	9
2.1	FireCCI41	9
2.2	FireCCISFD11	9
2.3	FireCCI50/1	10
2.4	FireCCILT10	12
3	Verification approach	12
3.1	Unit-level testing.....	12
3.2	Code sanity checks.....	12
3.3	Monitoring	13
3.4	Visual inspection.....	13
4	Data Acquisition Subsystem Verification.....	13
4.1	FireCCI50/1	13
4.2	FireCCILT10	14
5	Pre-Processing Subsystem Verification	15
5.1	Code sanity checks.....	15
5.1.1	FireCCI41	15

	Fire_cci System Verification Report		Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
			Issue	2.4	Date 11/01/2019
				Page	5

5.1.2	FireCCISFD11	15
5.2	Unit-level tests	15
5.2.1	FireCCI41	15
5.2.2	FireCCISFD11	18
5.3	Monitoring	18
5.3.1	FireCCI41	18
5.3.2	FireCCISFD11	18
5.4	Visual inspection.....	19
5.4.1	FireCCI41	19
5.4.2	FireCCISFD11	19
6	BA processing subsystem verification	20
6.1	Code sanity checks.....	20
6.2	Unit-level tests	20
6.2.1	FireCCI41	20
6.2.2	FireCCISFD11	21
6.2.3	FireCCI50/1	22
6.3	Monitoring	22
6.3.1	FireCCI41	22
6.3.2	FireCCISFD11	24
6.3.3	FireCCI50/1	24
6.3.4	FireCCILT10	25
6.4	Visual inspection.....	25
6.4.1	FireCCI41	25
6.4.2	FireCCISFD11	27
6.4.3	FireCCI50/1	28
6.4.4	FireCCILT10	31
7	Formatting subsystem verification	32
7.1	Code sanity checks.....	32
7.2	Unit-level tests	33
7.2.1	Generic tests	34
7.2.2	MERIS-specific tests	34
7.2.3	SFD-specific tests	36
7.2.4	MODIS-specific tests	37
7.2.5	AVHRR-specific tests	38
7.3	Calvalus monitoring.....	38
7.3.1	FireCCI41	38
7.3.2	FireCCISFD11	39
7.3.3	FireCCI50/1	39
7.3.4	FireCCILT10	40
7.4	Visual inspection.....	40
7.4.1	FireCCI41	41

7.4.2	FireCCISFD11	44
7.4.3	FireCCI50/1	45
7.4.4	FireCCILT10	49
Annex 1: Acronyms and Abbreviations		51
Annex 2: List of reference tiles and dates used for visual inspection of FireCCI41 products		52

List of Figures

Figure 2.1: Fire_cci Subsystems for MERIS processing.....	9
Figure 2.2: Fire_cci Subsystems SFD	10
Figure 2.3: Fire_cci MODIS processing cycle	11
Figure 2.4: Fire_cci Subsystems for MODIS processing	11
Figure 2.5: Fire_cci Subsystems AVHRR-LTDR.....	12
Figure 5.1: NDVI image of the daily composite of tile v08h20, 25 January 2008.	19
Figure 5.2: False-colour image of Sen2Cor result of granule 33PTK 18 Jan 2016.	20
Figure 6.1: BA output of tile v03h07, June 2008.	26
Figure 6.2: BA output of tile v08h20, January 2011.	26
Figure 6.3: BA of granule 32PKU, Feb 2016.....	27
Figure 6.4: Reference BA of granule 32PKU, Feb 2016.....	28
Figure 6.5: MODIS test tile location	29
Figure 6.6: BA in tile h08v05, Oct 2008	30
Figure 6.7: BA in tile h11v03, Apr 2008.....	30
Figure 6.8: BA in tile h30v10, Nov 2008	31
Figure 6.9: Global BA of FireCCILT10, July 2008.	31
Figure 6.10: Detail of Figure 6.9 for southern hemisphere Africa.	32
Figure 7.1: PSD compliant global BA product for June 2008.	41
Figure 7.2: PSD compliant BA product for January 2011, Africa.	41
Figure 7.3: PSD compliant global standard error for June 2008.....	42
Figure 7.4: PSD compliant global fraction of observed area for June 2008.	42
Figure 7.5: PSD compliant global number of patches for June 2008.....	42
Figure 7.6: PSD compliant BA in LC class 10 (cropland) for June 2008.	43
Figure 7.7: PSD compliant standard error product for January 2011, Africa.	43
Figure 7.8: PSD compliant fraction of observed area for January 2011, Africa.	43
Figure 7.9: PSD compliant number of patches for January 2011, Africa.	44
Figure 7.10: PSD compliant BA in LC class 10 (cropland) for January 2011, Africa...	44
Figure 7.11: Pixel product, Jan 2016, tile h36v16.....	45
Figure 7.12: Pixel product with artefact	45
Figure 7.13: PSD compliant global BA product of Dec 2004.....	46
Figure 7.14: PSD compliant global standard error of Dec 2004	46
Figure 7.15: PSD compliant fraction of burnable area of Dec 2004	46
Figure 7.16: PSD compliant global fraction of observed area of Dec 2004.....	47

Figure 7.17: PSD compliant global number of patches of Dec 2004	47
Figure 7.18: PSD compliant BA in LC class 10 (cropland) for Dec 2004	47
Figure 7.19: PSD compliant JD pixel product for March 2006, Africa. Grey pixels indicate unburnable areas, yellow pixels indicate unobserved areas, and red pixels indicate burned areas.	48
Figure 7.20: PSD compliant CL pixel product for March 2006, Africa for FireCCI50. It is clearly visible that different MODIS input tiles show different CL values. Still, the information correlates with the burned areas visible in the upper right of Figure 7.19.	48
Figure 7.21: PSD compliant CL pixel product for March 2006, Africa for FireCCI51. Although the colour scaling is identical, the values have changed, because the method for confidence level calculation has been improved. Also, the border effects are not visible anymore.	49
Figure 7.22: Detail of PSD compliant LC pixel product for March 2006, Africa. Black pixels indicate LC class 10, turquoise pixels indicate LC class 30, blue pixels indicate LC class 60, orange pixels indicate LC class 120, red colour indicates LC class 180. Occurrences of the respective pixels have been marked by respectively coloured circles for easier visibility.....	49
Figure 7.23: Burned Area, global, February 2000.....	50
Figure 7.24: Fraction of burnable area, February 2000	50
Figure 7.25: Fraction of observed area, February 2000	50

1 Introduction

1.1 Purpose of the document

This System Verification Report (SVR) documents the system verification activities of the ESA Climate Change Initiative (CCI) Fire project for its elements used in Phase 2 of the project. The system comprises the distributed subsystems for the burned area product generation with pre-processing, burned area retrieval, and formatting as described in the System Specification Document [SSD 2018].

Verification is the process to demonstrate that the system meets the specified requirements ([ECSS-E10 2009]), i.e. that the Fire_cci system is able to generate the burned area output products as specified in the Product Specification Document [PSD 2017] and to provide BA data to users. This comprises the identification of what shall be verified, the definition of how it shall be verified, the execution, and the reporting on pass or failure. Verification methods used are tests, inspection, and monitoring.


This version 2.3 of the document features the system verification of the system used to produce the MERIS and MODIS-based global data records (FireCCI41, FireCCI50, FireCCI51), of the system used to produce the first version of the Small Fires Dataset (FireCCISFD11), and of the system used to format the AVHRR LTDR BA data into global grid products (FireCCILT10).

1.2 Applicable Documents

[AD-1]	ESA Climate Change Initiative - CCI Project Guidelines. Ref. EOP-DTEX-EOPS-SW-10-0002, issue 1.0, date of issue 05 November 2010, available at http://cci.esa.int/filedepot_download/40/4
[AD-2]	ESA Climate Change Initiative (CCI) Phase 2 Statement of Work, prepared by ESA Climate Office, Reference CCI-PRGM-EOPS-SW-12-0012, Issue 1.3, date of issue 24 March 2015, available at http://www.esa-fire-cci.org/webfm_send/828
[DSWG 2015]	Data Standards Requirements for CCI data producers, CCI-PRGM-EOPS-TN-13-0009, Victoria Bennett and Sarah James, ESA Harwell, Issue 1.2, date of issue 24 March 2015, available at http://cci.esa.int/sites/default/files/CCI_Data_Requirements_Iss1.2_Mar2015.pdf
[ECSS-E10 2009]	Space engineering – Verification, ECSS-E-ST-10-02C, ESA ESTEC, Noordwijk, The Netherlands, 6 Mar 2009.

1.3 Reference Documents

[SSD 2018]	ESA CCI ECV Fire Disturbance: System Specification Document, T.Storm, M. Boettcher, G.Kirches, v1.5, date of issue 30 January 2018, available at http://www.esa-fire-cci.org/documents
[PSD 2017]	ESA CCI ECV Fire Disturbance: Product Specification Document, E. Chuvieco, M. L. Pettinari, A. Heil and T. Storm, Issue 6.3, date of issue 5 December 2017, available at http://www.esa-fire-cci.org/documents

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4		
		Issue	2.4	Date	11/01/2019
		Page			9

[O2.D2 2018]	G. Otón, E. Chuvieco (2018) ESA CCI ECV Fire Disturbance: O2.D2 Algorithm Theoretical Basis Document (ATBD) for AVHRR LTDR data, version 1.1. Available from: https://www.esa-fire-cci.org/documents
--------------	---

2 Systems under test

This section briefly introduces the different systems which are tested. All of these have been described in detail in [SSD 2018] and in [O2.D2 2018].

2.1 FireCCI41

Basically, the MERIS system consists of three independent subsystems: 1) the subsystem for pre-processing, 2) the subsystem for burned area retrieval, and 3) the subsystem for PSD-compliant formatting. The interfaces between these subsystems are intermediate files; see Figure 2.1 for an illustration.

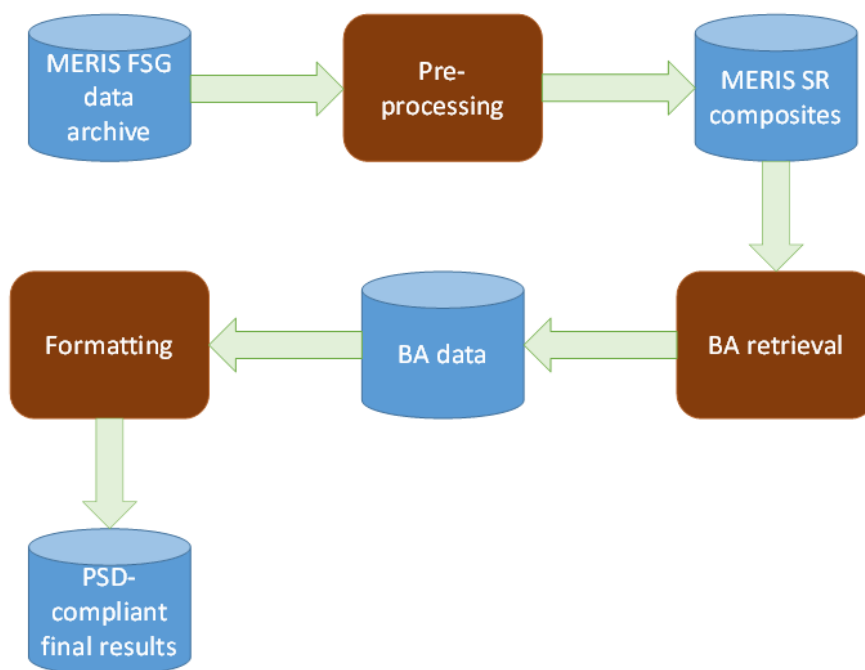


Figure 2.1: Fire_cci Subsystems for MERIS processing

The three subsystems of the MERIS Fire_cci system are verified independently; the methods of verification are adapted to the respective needs of the considered subsystem. This section lists the general methods which have been used for verification of the Fire_cci system. There are generally two different categories of verification: first, static verification of the code, second, verification that the results the code generates are correct. This second step includes the verification that the results are complete and show meaningful values in the expected range, defined by the algorithm developers.

2.2 FireCCISFD11

Similarly, the production system responsible for the production of the Small Fires Dataset based on Sentinel-2 (FireCCISFD11) is split into three independent subsystems, which use intermediate files as interfaces: 1) the subsystem for pre-processing, 2) the

subsystem for burned area retrieval, and 3) the subsystem for PSD-compliant formatting. See Figure 2.2 for an illustration.

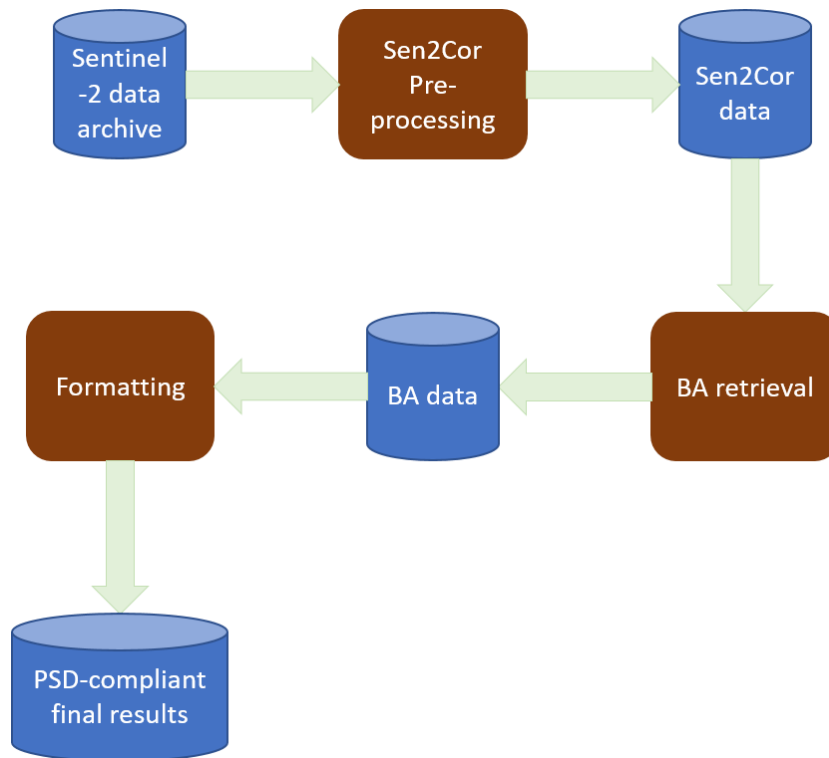


Figure 2.2: Fire_cci Subsystems SFD

The three subsystems of the SFD Fire_cci system are verified independently in the same way as the subsystems of the MERIS Fire_cci system.

2.3 FireCCI50/1

The MODIS processing system differs from the MERIS system and the SFD system in two key aspects: first, the process performed in Fire_cci starts from data which has been pre-processed by NASA, so no pre-processing is done within the MODIS system. Second, the BA retrieval is done on JASMIN instead of Calvalus. Third, more than half of the data has not been available from the start on the system, but needed to be acquired.

The formatting of the BA data has been done on Calvalus. Thus, the processing cycle depicted in Figure 2.3 was established: the data is acquired and processed by the MODIS BA processor, whose results are repackaged in order to create small, compressed files carrying exactly the information needed for the PSD creation, which is downloaded to Calvalus and formatted there.

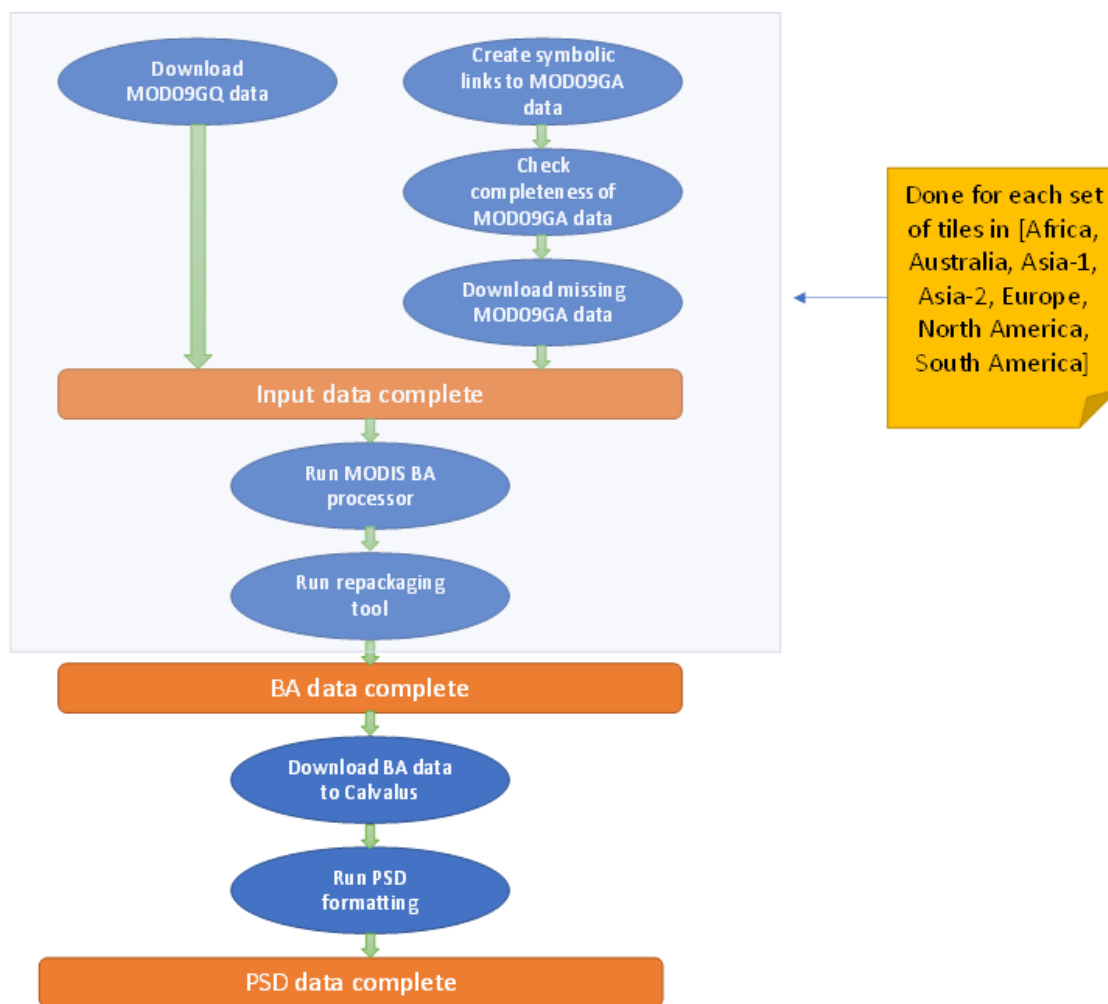


Figure 2.3: Fire_cci MODIS processing cycle

Consequently, the subsystems depicted in Figure 2.4 were established.

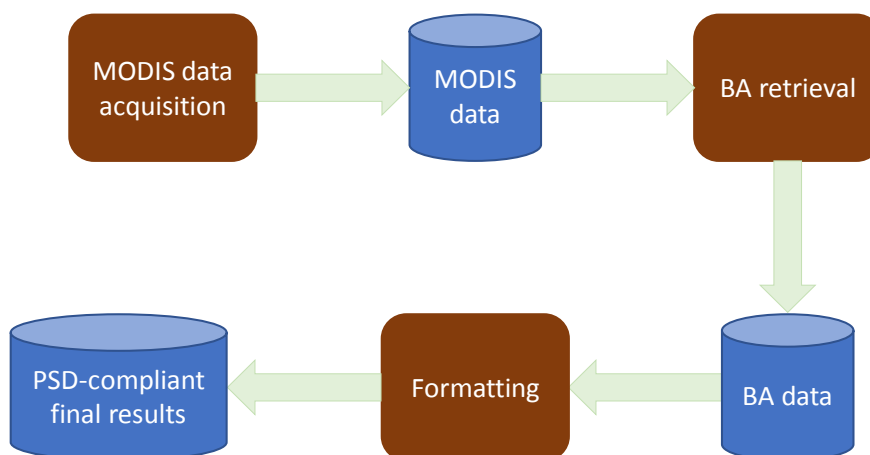


Figure 2.4: Fire_cci Subsystems for MODIS processing

2.4 FireCCILT10

The processing system for the AVHRR-LTDR Fire_cci v1.0 (FireCCILT10) dataset differs from the systems described in Section 2.1 to 2.3 in the main aspect that the burned area processing was done at UAH premises, while the formatting was done at BC. Due to the coarse resolution of the product, the size of the dataset was manageable with a standard Personal Computer, and was processed by the same team that developed the algorithm. The scheme depicted in Figure 2.5 was followed.

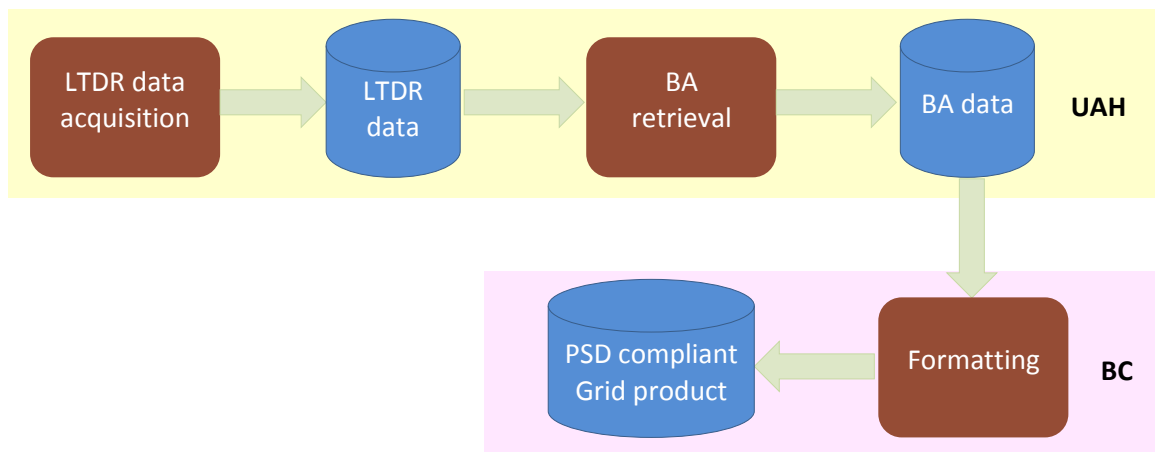


Figure 2.5: Fire_cci Subsystems AVHRR-LTDR

3 Verification approach

This section describes the generic verification approaches applied to the systems described in Section 2. The exceptions are the MODIS data acquisition subsystem and the AVHRR-LTDR subsystem, which are only tested for data completeness.

3.1 Unit-level testing

Unit-level testing is done during development of the system software, and before deploying it to operations. The principle of unit-level testing is to decompose the software into very small testable parts, the *units*, and to test each of the units. Testing in this case means that the expected result is defined before running the code, and to check afterwards if the actual result matches the expected result. If any test fails, the software is not deployed until the test succeeds.

Wherever possible, unit-level testing has been performed. The following sections, which show the actual verification, provide the results of the unit-level testing that has been performed.

Since this verification method is applied before deploying the code, it falls into the first category of verification.

3.2 Code sanity checks

The purpose of code sanity checks is to verify at runtime that the state of processing is fine. Typically, input/output data, user-provided configuration parameters, computational results and many more may be validated. If the validation fails, there are different possibilities on how to deal with the situation: it may be considered potentially harmful, which typically results in logging a warning, which in turn is followed by visual inspection, or it may be considered severe, which typically stops the process.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		13

Code sanity checks are designed principally, but not solely, to identify programming errors which are only to be found at runtime; for example, if a routine expected to compute some absolute value yields a negative value, there must be a programming error.

If applicable, the following sections will show lists of the code sanity checks the respective subsystems are running.

3.3 Monitoring

Monitoring is twofold: first, it is performed during runtime. As all three Fire_cci subsystems are integrated in Calvalus, the computing cluster at Brockmann Consult, the Calvalus monitoring applies to all three of them. In the MODIS case, only the PSD formatting step is done on Calvalus, whilst the same principle of monitoring is done on JASMIN for the BA production.

All three subsystems are controlled by the software package *pmonitor*, which allows the creation of sub-tasks and tracks them. The typical workflow looks like this: a job, which consists of multiple sub-tasks, is started and submitted to Calvalus. If a subtask fails, it is re-tried on a different physical machine in the cluster, up to a configurable count. If the subtask still fails, the error is reported back to *pmonitor*, which adds the job as a failed job to a report file.

This technique allows to apply the fix, so the failed job can run fine, and re-run only the failed jobs without having to care for an extra list of failed jobs manually. Hence, the check to verify that the system has computed a complete output data set is to inspect the report files, and look for failed jobs.

Second, monitoring also means to check the file output paths for completeness. It is possible to roughly compute how many files are expected, and to compare this number with the number of actual files.

3.4 Visual inspection

Visual inspection, as the last verification step, means to download the results, open them with a suitable viewer software (depending on the purpose, this might be ncdump, Panoply, ArcGIS, SNAP or similar), and have a look at the images. Moreover, the values of arbitrarily chosen pixels have been compared to the same pixels (i.e. the pixels at the same geo-location) of the previously created images.

This is an important step to verify that the results are sane; errors in geo-information, scaling, tiling, data distribution and many more can easily be found with visual inspection.

4 Data Acquisition Subsystem Verification

4.1 FireCCI50/1

This step verifies the completeness of the input dataset for the MODIS BA retrieval. As described in [SSD 2018], there are basically two input datasets: MOD09GQ, and MOD09GA. The MOD09GQ data was fully downloaded to the system, while the MOD09GA data was mostly there already.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		14

First, a list of expected files which should exist on the system is created. Second, the actually existing files are collected into a second list. Third, the lists are compared to each other; if they differ, correctional steps have to be applied.

Two dedicated routines have been put into place, which verify the completeness of these two datasets. Both consist of a script identifying the missing images, and of a script to fill the gaps. These scripts work as documented in the following pseudo-code:

```
for each year y of the time series, do:
  for each input tile t, do:
    for each day day in year y, do:
      if MODIS file for tile t, year y and day day does not exist:
        write an entry to file missing-ga.out

for all entries in file missing.out, do:
  ask NASA web server if file should exist
  if file should exist:
    download and check against checksum
  else:
    mark missing file as acceptable
```

4.2 FireCCILT10

The database files were checked one by one with a code created to find errors in the images, such as gap days, corrupt HDF or band, repeated images in the same day and sensor. First, it was checked that all dates had files. Second, the existing files were opened and the corrupt files were discarded.

The pseudo-code used for this step is:

```
for all dates (daily, 1982-2017):
  count the number of files on this day
  if there is a file:
    read file
    write the name of the file
  elif there are files:
    if the sensor is different:
      write file names
    else:
      the files are repeated
      write and delete the file with the first
processing date
      write the name of the file with the last
processing date
  else:
    write the name of the missing file
for existing files:
```

 fire cci	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
			Page	15

```

try:
    open HDF
    for each band in HDF:
        try:
            open band
        except error:
            write corrupt band and name of HDF
            delete corrupt file
except:
    write corrupt HDF
    delete corrupt file

```

5 Pre-Processing Subsystem Verification

This section shows the verification for the MERIS and the SFD pre-processing subsystems. There is no pre-processing for MODIS or AVHRR performed within the scope of the Fire_cci system, as the data is downloaded already pre-processed. In the MERIS pre-processing system, unit-level testing, code sanity checks, monitoring and visual inspection have been performed; in the SFD pre-processing system, as it employs third-party software¹, only monitoring and visual inspection have been performed.

5.1 Code sanity checks

5.1.1 FireCCI41

A wide number of code sanity checks have been introduced into the pre-processing code; it is out of the scope of this document to list them all, so some hand-picked examples are provided below. The SDR retrieval process fails if:

- some error occurs while loading the auxiliary data
- the elevation model (e.g. GETASSE) cannot be found
- the source product does not have geo-information
- the source product is missing a band
- the computed sun angles are not smaller or equal to 85.0°

5.1.2 FireCCISFD11

Describing the code sanity checks performed within the third-party software Sen2Cor is out of the scope of this document. However, the code is open source², so the code sanity checks done within that code are available for inspection.

5.2 Unit-level tests

5.2.1 FireCCI41

The MERIS pre-processing subsystem has been extensively unit-level tested. See below for the report. Note that the pre-processing software is part of a larger software bundle,

¹ Sen2Cor, see <http://step.esa.int/main/third-party-plugins-2/sen2cor/>

² <https://github.com/umwilm/SEN2COR/tree/master/sen2cor>

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
			Page	16

so only those tests relevant to the actual Fire_cci pre-processing are included in the report.

Test the NetCDF-Writer:

```
Test set:
org.esa.beam.globalbedo.inversion.io.netcdf.AlbedoNc4WriterLoad
dAsServiceTest
```

```
-----
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.085 sec - in
org.esa.beam.globalbedo.inversion.io.netcdf.AlbedoNc4WriterLoad
dAsServiceTest
```

Test that spectral albedo bands are written correctly:

```
Test set:
org.esa.beam.globalbedo.inversion.spectral.SpectralAlbedoTest
```

```
-----
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.031 sec - in
org.esa.beam.globalbedo.inversion.spectral.SpectralAlbedoTest
```

Test spectral inversion code:

```
Test set:
org.esa.beam.globalbedo.inversion.spectral.SpectralInversionTest
```

```
-----
Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.053 sec - in
org.esa.beam.globalbedo.inversion.spectral.SpectralInversionTest
```

Test generic utils for albedo inversion computation:

```
Test set:
org.esa.beam.globalbedo.inversion.util.AlbedoInversionUtilsTest
```

```
-----
Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.601 sec - in
org.esa.beam.globalbedo.inversion.util.AlbedoInversionUtilsTest
```

Test generic I/O utility code:

```
Test set: org.esa.beam.globalbedo.inversion.util.IOUtilsTest
```

```
-----
Tests run: 12, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.216 sec - in
org.esa.beam.globalbedo.inversion.util.IOUtilsTest
```

Test monthly weighting of albedos:

```
Test set:
org.esa.beam.globalbedo.inversion.util.MonthlyAlbedoTest
```

```
-----
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.053 sec - in
org.esa.beam.globalbedo.inversion.util.MonthlyAlbedoTest
```

Test the SVD code:

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		17

Test set:
org.esa.beam.globalbedo.inversion.util.SingleValueDecompositionTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.063 sec - in
org.esa.beam.globalbedo.inversion.util.SingleValueDecompositionTest

Test some generic utility code for Broadband Directional Reflectance computation:

Test set: org.esa.beam.globalbedo.bbdr.BbdrUtilsTest

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.085 sec - in org.esa.beam.globalbedo.bbdr.BbdrUtilsTest

Test the lookup table for gaseous corrections:

Test set: org.esa.beam.globalbedo.bbdr.GasLookupTableTest

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.138 sec - in org.esa.beam.globalbedo.bbdr.GasLookupTableTest

Test some meta information of single pixel processor:

Test set:
org.esa.beam.globalbedo.bbdr.GlobalbedoLevel2SinglePixelTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.085 sec - in
org.esa.beam.globalbedo.bbdr.GlobalbedoLevel2SinglePixelTest

Test different look-up tables for MERIS processing:

Test set: org.esa.beam.globalbedo.bbdr.MerisLutTest

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.855 sec - in org.esa.beam.globalbedo.bbdr.MerisLutTest

Test azimuth and zenith angles computation:

Test set: org.esa.beam.globalbedo.bbdr.MeteosatGeometryTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.054 sec - in org.esa.beam.globalbedo.bbdr.MeteosatGeometryTest

Test NetCDF writer of bbdr data:

Test set:
org.esa.beam.globalbedo.bbdr.netcdf.BbdrNc4WriterLoadedAsServiceTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.153 sec - in
org.esa.beam.globalbedo.bbdr.netcdf.BbdrNc4WriterLoadedAsServiceTest

Test operator for land-cover quality analysis:

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		18

Test set: org.esa.beam.landcover.LcQaOpTest

Tests run: 6, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.786 sec - in org.esa.beam.landcover.LcQaOpTest

Test writing of geocoding:

Test set: org.esa.beam.landcover.ScripGeocodingWriterTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.935 sec - in org.esa.beam.landcover.ScripGeocodingWriterTest

Test cloud detection:

Test set: org.esa.beam.landcover.UclCloudDetectionTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.069 sec - in org.esa.beam.landcover.UclCloudDetectionTest

5.2.2 FireCCISFD11

Describing unit-level testing performed within the third-party software Sen2Cor, if existing, is out of the scope of this document.

5.3 Monitoring

5.3.1 FireCCI41

The report file shows the following:

```
thomas@master01:~/fire-inst/meris-attic $ cat l23.status
495 created, 0 running, 0 backlog, 495 processed, 0 failed
```

This looks reasonable: the l23 consists of the steps “SDR”, “SR”, and “nccopy”, where SDR is run for each month, SR is run for each 10 days, and nccopy is run for each year. So the expected output is $10 * 12 + 10 * 365/10 + 10$ (10 years times each month plus 10 years times the number of days in a year divided by 10, plus 10 years of nccopy) = 495.

5.3.2 FireCCISFD11

The report file for the Sentinel-2 pre-processing shows the following:

```
thomas@master01:~/fire-inst $ cat s2.status
4720 created, 0 running, 0 backlog, 4720 processed, 0 failed
```

The granularity of the pre-processing is tile-based. Note that as Sentinel-2 calls its tiles ‘granules’, this term will be further used in this document. There are 2360 different granules needed to create the SFD. As the pre-processing step is twofold, consisting of the run of Sen2cor and an additional potential merging of same-granule inputs, the number of 4720 successfully processed requests is the expected one ($= 2 * 2360$).

5.4 Visual inspection

5.4.1 FireCCI41

There have been a number of random sample checks; Figure 5.1 shows an example for one of the inspected files. For verification, this and similar images have been sent to the algorithm developers in GeoTiff format; the algorithm developers verified successfully that the values are within the expected range and that there are no obvious shifts in the geo-location. The verification has been performed in the way that the algorithm developers created a hidden reference file before, and compared it to the results, in order to allow for a non-biased verification. No substantial deviations have been detected.

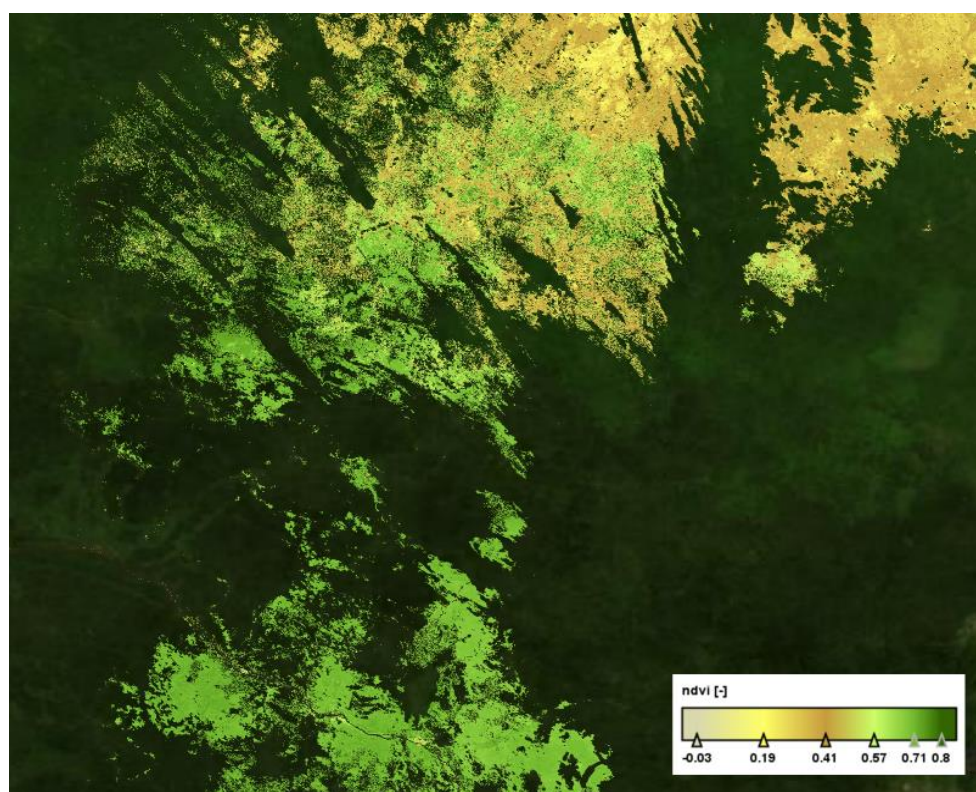


Figure 5.1: NDVI image of the daily composite of tile v08h20, 25 January 2008.

5.4.2 FireCCISFD11

Similarly, visual inspection has been done with the Sentinel-2 pre-processing results. Multiple files have been randomly checked by the system administrator and by the algorithm developers; the algorithm developers verified successfully that the values are within the expected range and that there are no obvious shifts in the geo-location. The verification has been performed in the way that the algorithm developers created a hidden reference file before, and compared it to the results, in order to allow for a non-biased verification. No substantial deviations have been detected. See Figure 5.2 for an example image.

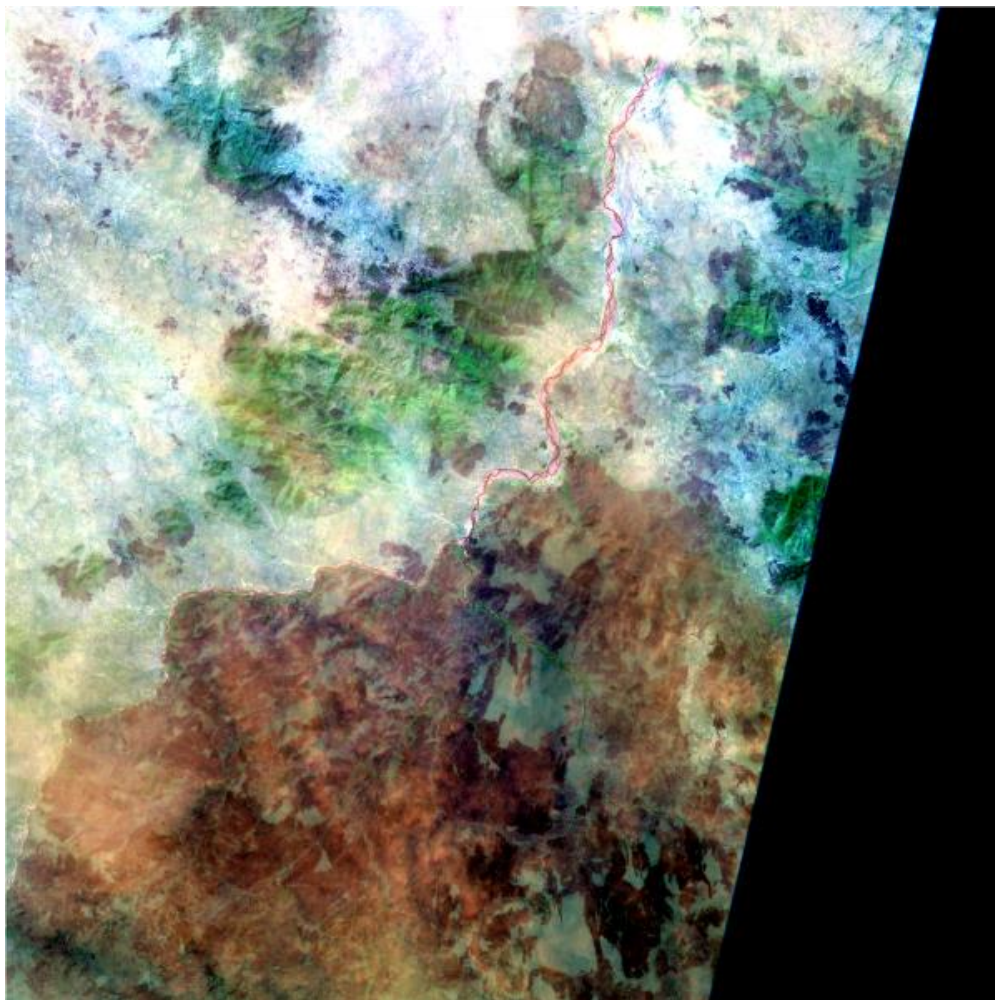


Figure 5.2: False-colour image of Sen2Cor result of granule 33PTK 18 Jan 2016.

6 BA processing subsystem verification

6.1 Code sanity checks

The code sanity checks in the BA processing subsystems of MERIS, MODIS, and the SFD are intentionally simple: if anything goes wrong, the process is stopped and an error message as well as all intermediate data is output.

This happens, for example, if the input data is corrupt or the auxiliary data files cannot be found.

As the following sections show, these errors have not occurred in the final processing of the data set.

6.2 Unit-level tests

6.2.1 FireCCI41

The BA processing software has been unit-level tested by the algorithm developers. The unit-level testing has been performed by running scripts which compared the results of the algorithm for controlled areas with measurements for those areas taken before.

6.2.2 FireCCISFD11

The BA processing software is a Java re-implementation of the original implementation, which was done in Python. In order to ensure that correct results are created, a large number of JUnit³ tests have been successfully performed. See below for the output of the test reports:

Test the reader code of the active fires:

```
Test set: org.esa.cci.fire.s2ba.ActiveFireReaderTest
-----
Tests run: 2, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.593 sec
```

Test the ccl code:

```
Test set: org.esa.cci.fire.s2ba.ConnectedComponentLabelingTest
-----
Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 sec
```

Test the convolution code:

```
Test set: org.esa.cci.fire.s2ba.ConvoluterTest
-----
Tests run: 3, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.442 sec
```

Test the dilation code:

```
Test set: org.esa.cci.fire.s2ba.DilaterTest
-----
Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 sec
```

Test the hole removal code:

```
Test set: org.esa.cci.fire.s2ba.HoleRemoverTest
-----
Tests run: 7, Failures: 0, Errors: 0, Skipped: 3, Time elapsed:
0.001 sec
```

Test the S2 Burned Area algorithm:

```
Test set: org.esa.cci.fire.s2ba.S2BaAlgorithmTest
-----
Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0 sec
```

Test the finalising BA mapper code:

```
Test set: org.esa.cci.fire.s2ba.S2BaPostMapperTest
-----
Tests run: 10, Failures: 0, Errors: 0, Skipped: 7, Time elapsed:
1.15 sec
```

³ <http://junit.org/>

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		22

Test the mask code:

```
Test set: org.esa.cci.fire.s2ba.SclMaskTest
```

```
-----
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.963 sec
```

Test the generic tool code:

```
Test set: org.esa.cci.fire.s2ba.ToolsTest
```

```
-----
Tests run: 7, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec
```

Note that the large number of “skipped” tests does not imply any cover-up of failures; rather, these tests have been used to create intermediate result files for visual inspections, and are therefore skipped in the production chain.

6.2.3 FireCCI50/1

The BA retrieval code has not been unit-level tested. It has been written by scientists, who are not familiar with the concept of unit-level testing. However, algorithm developers have analysed the special scenarios where the code is supposed to work. Apart from the study sites (MODIS tiles) used for the theoretical development of the algorithm, other tiles were processed in order to ensure the proper performance of the code. So a test sample was designed selecting those tiles that represent a special case. Thanks to that test sample all the expected errors were handled by adding conditional code fragments (example below), but also try/exception blocks. Those special cases include missing images, no HS available, corrupt image, etc.

```
if no_images:
    print "No images have been found in the file system"
    sys.exit()
```

Besides, the special cases that were not expected at the beginning progressively appeared during the test processing carried out during the implementation of the code at the System Engineers’ premises. Those unexpected errors were properly handled in the algorithm code. As a consequence, the final version of the code reported no errors related to the code itself and was successfully processed globally for the whole time series.

6.3 Monitoring

6.3.1 FireCCI41

The status after running all BA retrieval jobs:

```
thomas@master01:~/fire-inst/meris-attic $ cat fire.status
4440 created, 0 running, 0 backlog, 4440 processed, 0 failed
```

This shows that all BA retrieval jobs have run successfully.

Also, this tells the operator that the expected count of result files is 4440; since there may be several tif files, we rather count the log files:

```
thomas@feeder01:/calvalus/projects/fire/meris-ba $ find . -name
"*BA-v4.05.log" | wc -l
```

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		23

4440

There has also been an extra verification step added, as the process sometimes failed while claiming it did not fail for an unknown reason – this was developed because the number of result files was observed to be less than expected. In such cases, a file entry has been made to a dedicated “error”-directory, outputting the intermediate data (“auxdata”) for investigation. Hence, the check if everything has been computed correctly is to a) either check if the error-directory is empty, or, if it is not, that for each entry in the error directory there is a later result file.

```
thomas@feeder01:/calvalus/projects/fire/errors$ ls -ltr
-rw-rw-r-- 1 yarn bc 199603161 Jun 10 23:36 auxdata-2011-
v05h30.tar.gz
-rw-rw-r-- 1 yarn bc 1788310194 Jun 10 23:40 auxdata-2011-
v11h32.tar.gz
```

This shows that there have been two erroneous processing runs, so the result files’ timestamp must be checked:

```
thomas@feeder01:/calvalus/projects/fire$ ls -l meris-
ba/2011/*v05h30*.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 00:07 meris-
ba/2011/BA_PIX_MER_v05h30_201104_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 00:07 meris-
ba/2011/BA_PIX_MER_v05h30_201105_v4.1.tif

thomas@feeder01:/calvalus/projects/fire$ ls -l meris-
ba/2011/*v11h32*.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201101_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201102_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201103_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201104_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201105_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201106_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201107_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201108_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201109_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201110_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201111_v4.1.tif
-rw-rw-r-- 1 thomas bc 155549260 Jun 11 02:51 meris-
ba/2011/BA_PIX_MER_v11h32_201112_v4.1.tif
```

The timestamps of all result files are later than the timestamp of the files indicating the error, so they have been successfully re-processed.

6.3.2 FireCCISFD11

The status of the processing subsystem after running all BA retrieval jobs of the SFD shows:

```
thomas@master01:~/fire-inst $ cat s2-ba.status
2360 created, 0 running, 0 backlog, 2360 processed, 0 failed
```

This shows that all 2360 BA retrieval jobs have run successfully – one for each considered granule:

```
thomas@master01:~/fire-inst $ cat s2-tiles.txt | wc -l
2360
```

Also, in order to check that data has been produced, it is possible to count the produced burned-area files per granule:

- 1) Count the retrieved BA images per granule and store the result:

```
for t in $(ls -d T*) ; do echo "#BA in tile $t=$(ls $t/BA* | wc -l)" ; done > BA-per-tile.out
```

- 2) Check that 0 tiles with a BA image count of 0 are existing:

```
grep =0 BA-per-tile.out | wc -l
0
```

6.3.3 FireCCI50/1

Running BA retrieval jobs creates status files. Thus, for each continental cycle there is a status file. The number of successfully processed jobs should match the number of years to process (2000 – 2016 = 17)⁴ times the number of MODIS tiles in that continental zone:

Continental zone	# tiles	# processing jobs
Africa	41	697
Asia	69	1173
Australia	34	578
Europe	15	255
North America	38	646
South America	58	986

```
cat modis-africa.status
697 created, 0 running, 0 backlog, 697 processed, 0 failed
cat modis-asia.status
1173 created, 0 running, 0 backlog, 1173 processed, 0 failed
cat modis-australia.status
```

⁴ Note that although the data of year 2000 has been processed, the resulting images are empty.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		25

```

578 created, 0 running, 0 backlog, 578 processed, 0 failed
cat modis-europe.status
255 created, 0 running, 0 backlog, 255 processed, 0 failed
cat modis-namerica.status
646 created, 0 running, 0 backlog, 646 processed, 0 failed
cat modis-samerica.status
986 created, 0 running, 0 backlog, 986 processed, 0 failed

```

This shows that all BA retrieval jobs have run successfully.

6.3.4 FireCCILT10

The code checks that the files were created or the programme stops. It counted the existing files and verified that they were correct (36 years x 12 months = 432 files) for the whole time series for each of the intermediate and final products. In the case of 1994, in the months when there is no data, the files were still created, but with empty data.

```

for year in the time series:
for month in the year:
    if the file exist:
        write file name
        file number +=1
    else:
        stop

```

6.4 Visual inspection

6.4.1 FireCCI41

Some dedicated samples have been visually inspected, as well as some random samples. For dedicated checks, tile v03h07 of June 2008 located in Northern Canada was chosen by the algorithm developers. This particular region has been chosen because it is a boreal forest region in which small fires (< 200 ha) predominate, although large fires account the great part of the total BA. The perimeters are provided by the Canadian National Fire Database (CNFDB). The result image (configured to show only BA with a slightly transparent world map in the background, with – arbitrarily chosen – different colours indicating different days of burn) is shown in Figure 6.1. The verification has been performed in the way that the algorithm developers created a hidden reference file before, and compared it to the results, in order to allow for a non-biased verification. No substantial deviations have been detected.

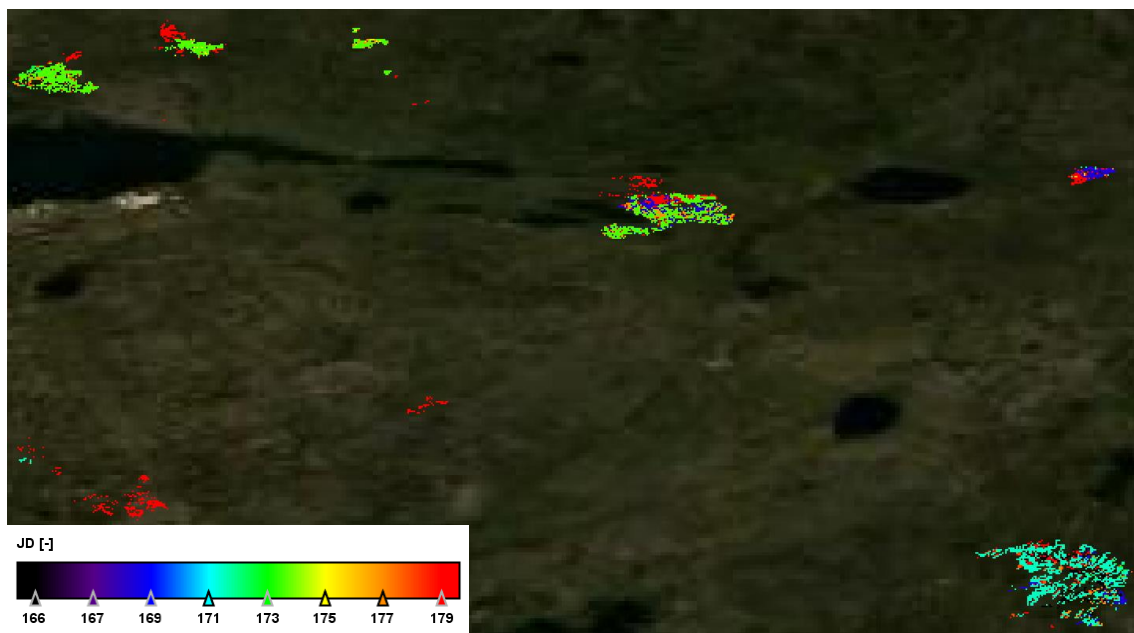


Figure 6.1: BA output of tile v03h07, June 2008.

Also, some other random samples have been chosen to be inspected, mostly in African tiles where there are many burns, such as tile v08h20 at January 2011. The selection of date and location was made by the algorithm developers, based on previously gathered experience. The result image is shown in Figure 6.2 (configured to show only BA with a slightly transparent world map in the background, with – arbitrarily chosen – different colours indicating different days of burn).

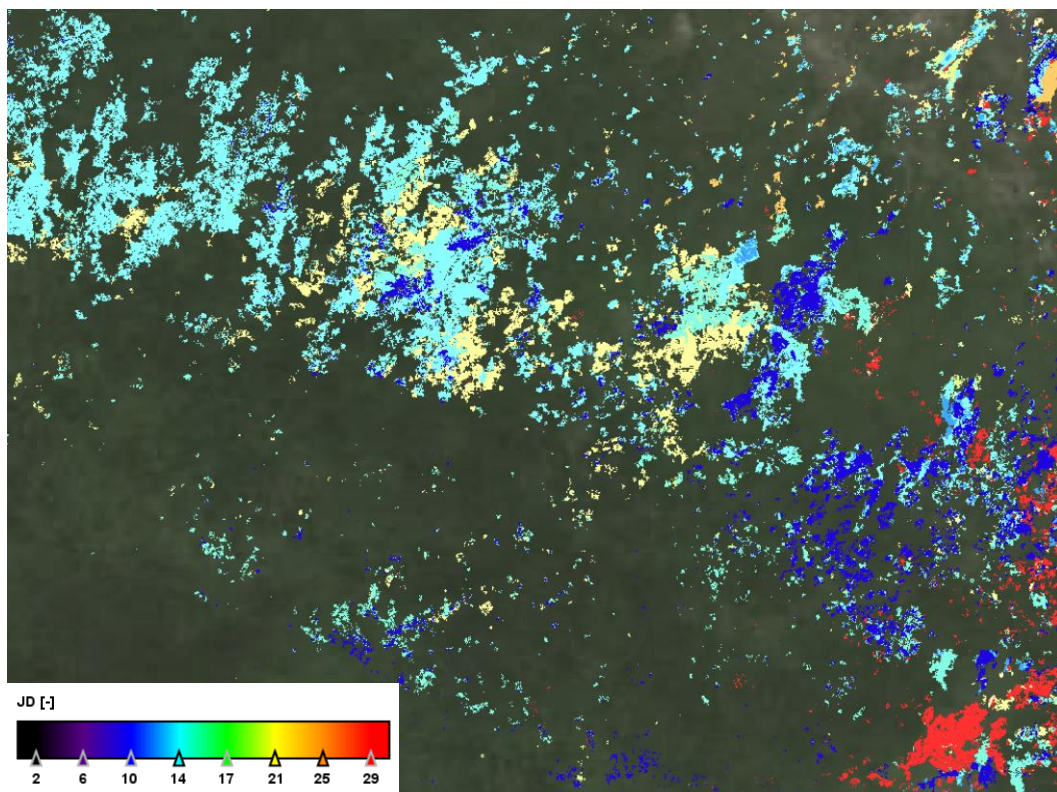


Figure 6.2: BA output of tile v08h20, January 2011.

Note that these are examples for burned area only, while there have been visual inspection checks also covering other parameters. The other tests performed to the random sampled data (all files tested are listed in Annex 2) were:

- The number and location of the pixels classified as burned must be the same as the ones calculated by the algorithm developers when running the algorithm at their premises.
- Value range: the values of layers must be $0 \leq \text{day of burn} \leq 366$; confidence level ≥ 0 ; $0 \leq \text{land cover class} \leq 180$, and they must be equal to the values calculated by the algorithm developers when running the algorithm at their premises.
- Geographic extent of each file must correspond to the coordinates specified in [PSD 2017].

6.4.2 FireCCISFD11

Multiple sample granules were defined by the algorithm developers; visual inspection has been done on these granules. These 49 tile samples were chosen due to the different land covers they contain (from tree covers and shrublands to croplands and urban areas, according to the ESA CCI Land Cover map of year 2015) and the large burned surface in the 2015-2016 fire season with a high variability on fire sizes (according to the SFD).

Figure 6.3 shows an example result image of BA retrieved data; the Sentinel-2 granule is 32PKU (at around 12° N 6.5° W). The verification has been performed in the way that the algorithm developers created that hidden reference file before, and compared it to the results, in order to allow for a non-biased verification. No substantial deviations have been detected. An image of the hidden reference file is provided in Figure 6.4; the slight differences have been classified as acceptable by the algorithm developers, as different libraries have been used between the production code, which is written in Java, and the development code, which has been delivered in Python.

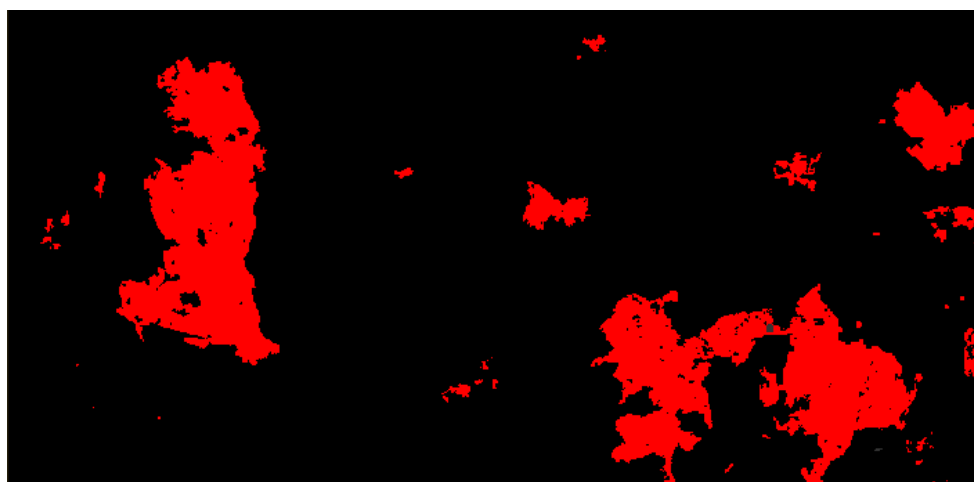


Figure 6.3: BA of granule 32PKU, Feb 2016

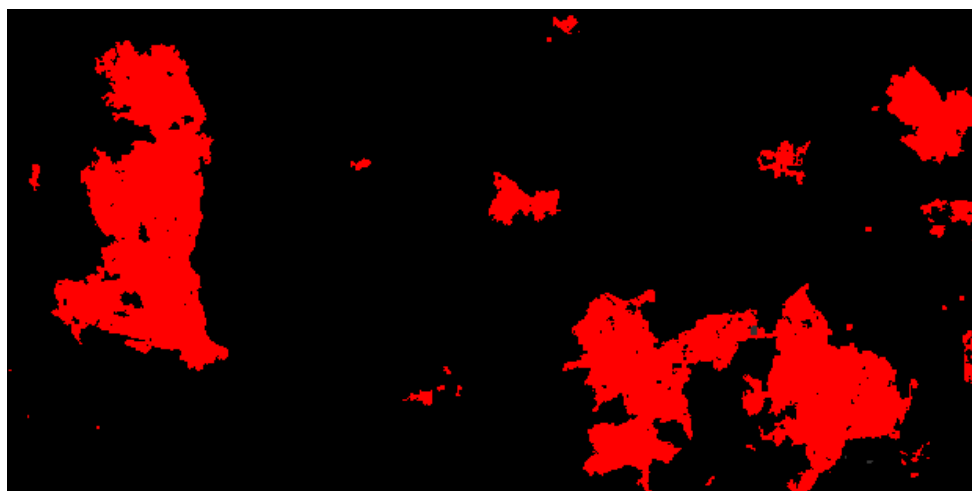


Figure 6.4: Reference BA of granule 32PKU, Feb 2016

This is only an example; many other dates have been chosen for inspection, covering the 49 granules listed in Table 1. This list, as well as the selection of dates has been made by the algorithm developers, based on previously gathered experiences. Before producing the final dataset, the algorithm developers confirmed for all of the files that have been visually inspected that they look as expected in terms of value range, pixel distribution, and geographic extent.

Table 1: Sentinel-2 BA sample granules

30NYL	30PYS	31NCH	31NFF	31PDK
30NYM	30PZQ	31NCJ	31NFG	31PDL
30NYN	30PZR	31NDF	31NFH	31PDM
30NYP	30PZS	31NDG	31NFJ	31PEK
30NZL	31NBF	31NDH	31PBK	31PEL
30NZM	31NBG	31NDJ	31PBL	31PEM
30NZN	31NBH	31NEF	31PBM	31PFK
30NZP	31NBJ	31NEG	31PCK	31PFL
30PYQ	31NCF	31NEH	31PCL	31PFM
30PYR	31NCG	31NEJ	31PCM	

6.4.3 FireCCI50/1

Three test tiles have been identified for visual inspection: h08v05, h11v03, and h30v10. See Figure 6.5 for the location of these tiles on a world map.

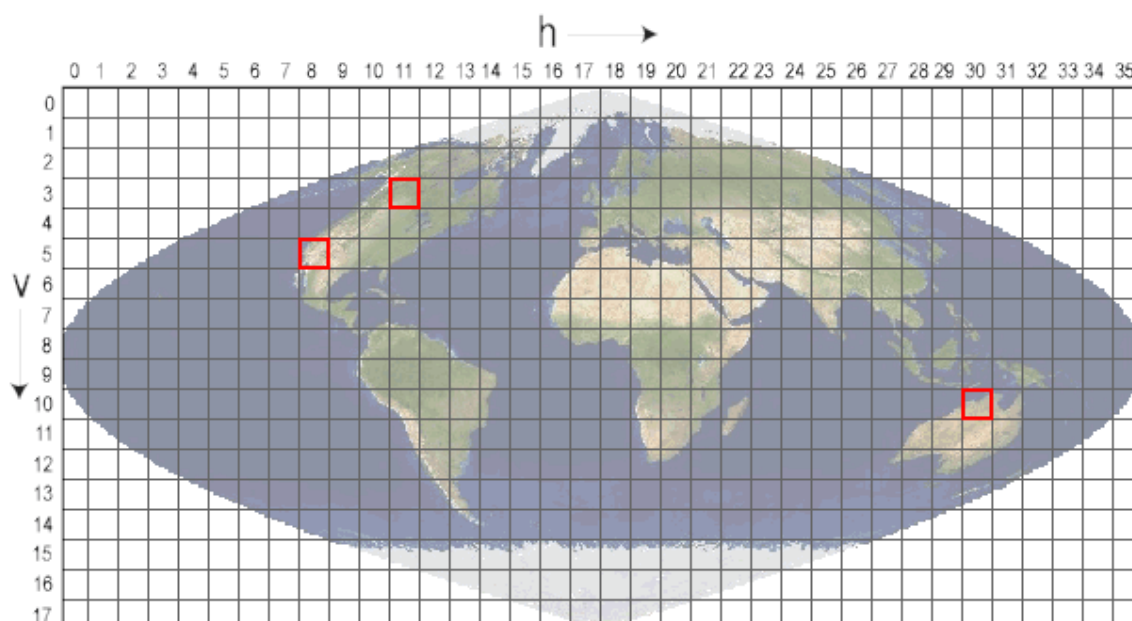


Figure 6.5: MODIS test tile location

Tile h30v10 is located in the North of Australia, which is a tropical savanna region that is characterised for its high fire occurrence and large fires. The North Australian Fire Information (NAFI) provides the perimeter data.

Tile h11v03 covers a boreal forest region, see Section 6.4.1 for the justification.

In tile h08v05, Mediterranean and temperate forests and also shrublands are mainly found. It is a relatively high fire occurrence region. Fire and Resource Assessment Program (FRAP) of California provides state level fire perimeters.

For these three tiles, the whole year 2008 has been processed for inspection. Examples of resulting images (configured to show BA in red, non-burned area in green, non-burnable areas in blue, and unobserved areas in yellow, the latter corresponding to pixels with sensor failure, clouds and cloud shadows(extracted from the quality flags of the input data)) are shown in Figure 6.6 - Figure 6.8. The verification has been performed in the way that the algorithm developers created a hidden reference file before, and compared it to the results, in order to allow for a non-biased verification. No deviations have been detected.

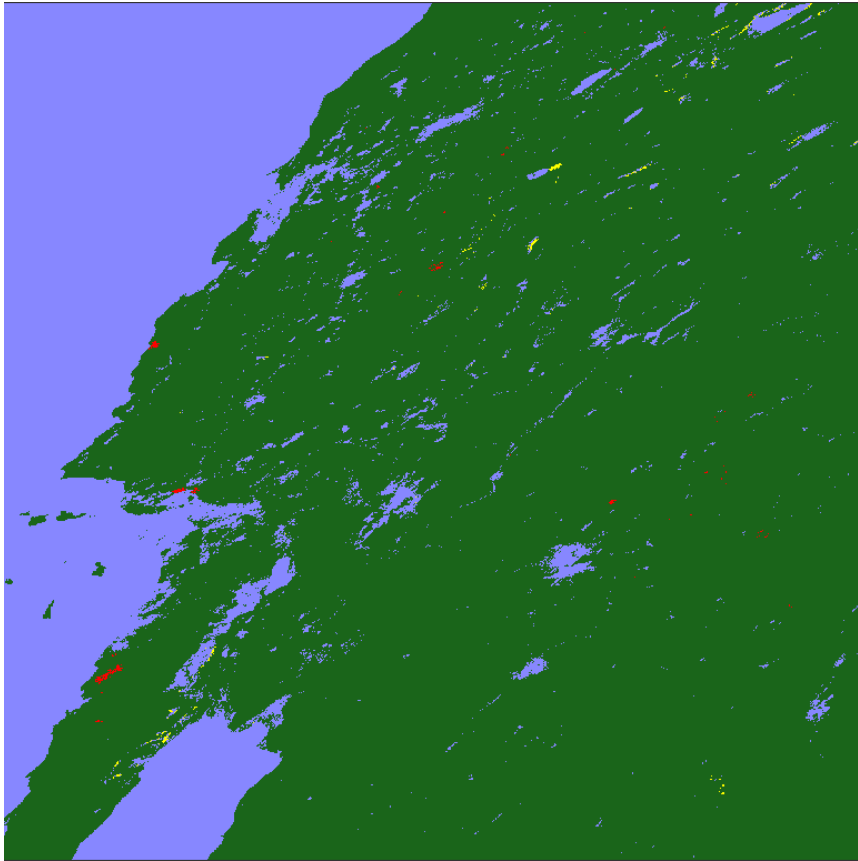


Figure 6.6: BA in tile h08v05, Oct 2008

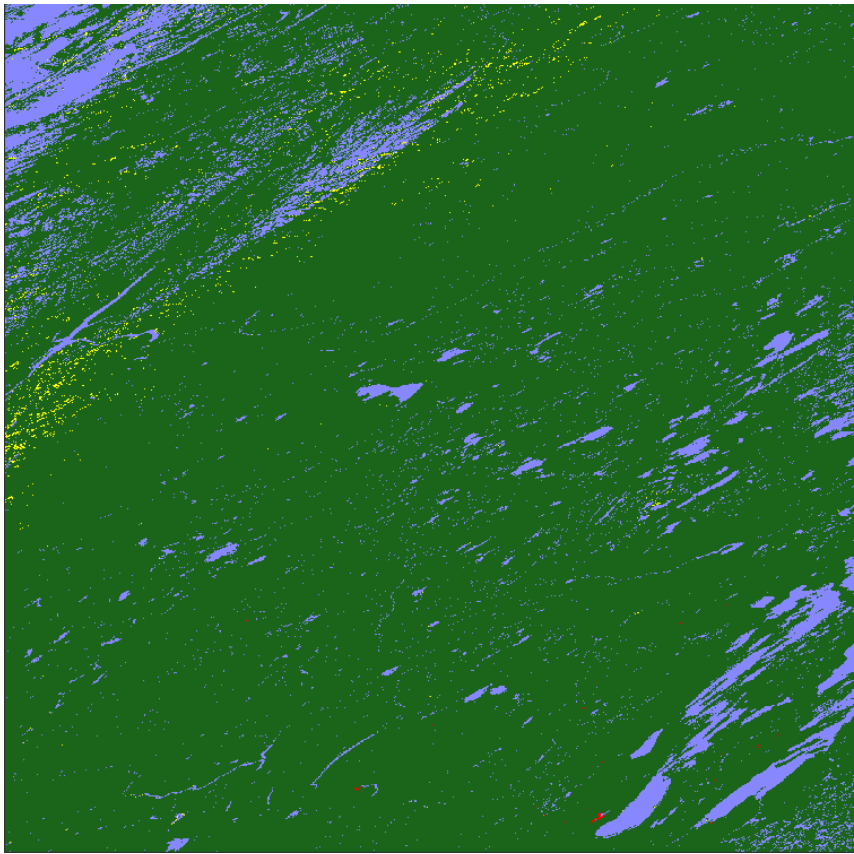


Figure 6.7: BA in tile h11v03, Apr 2008

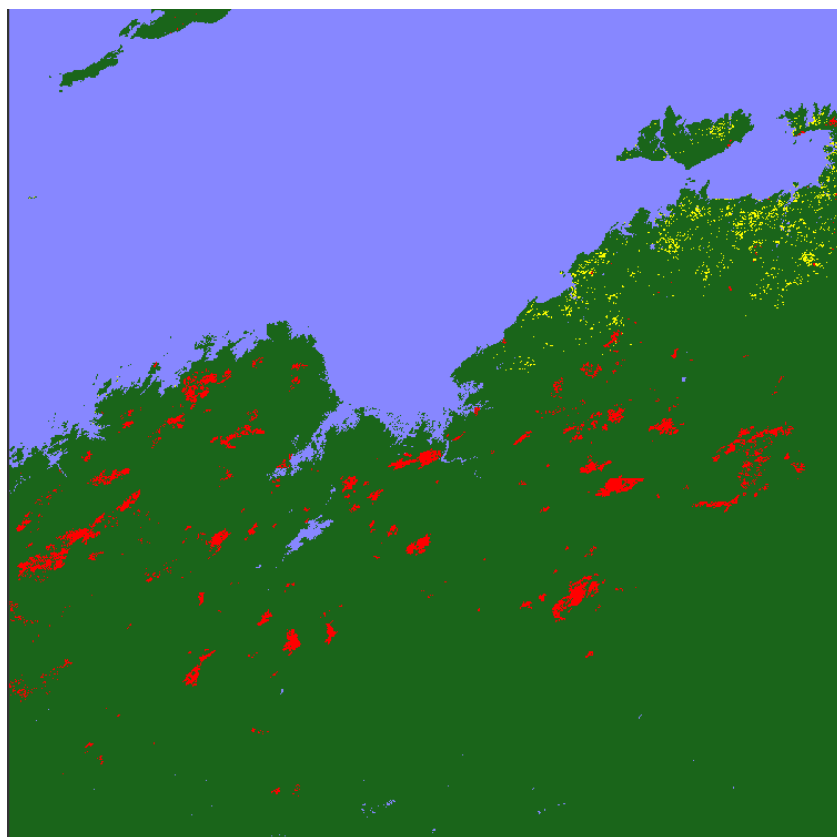


Figure 6.8: BA in tile h30v10, Nov 2008

6.4.4 FireCCILT10

All burned area images were visually checked by the algorithm developers to verify they had data. They were opened manually in QGIS and the data was checked (with a color palette of blue-yellow-red from lowest to highest BA). The presence of data was verified, as well as its coherence with other existing BA products (Figure 6.9 and Figure 6.10)

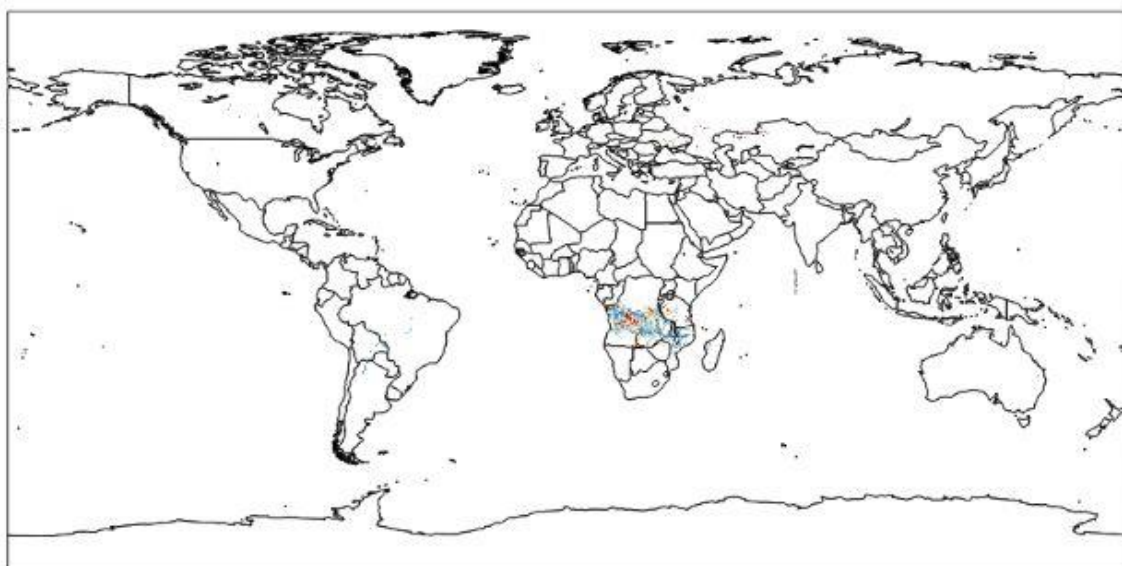


Figure 6.9: Global BA of FireCCILT10, July 2008.

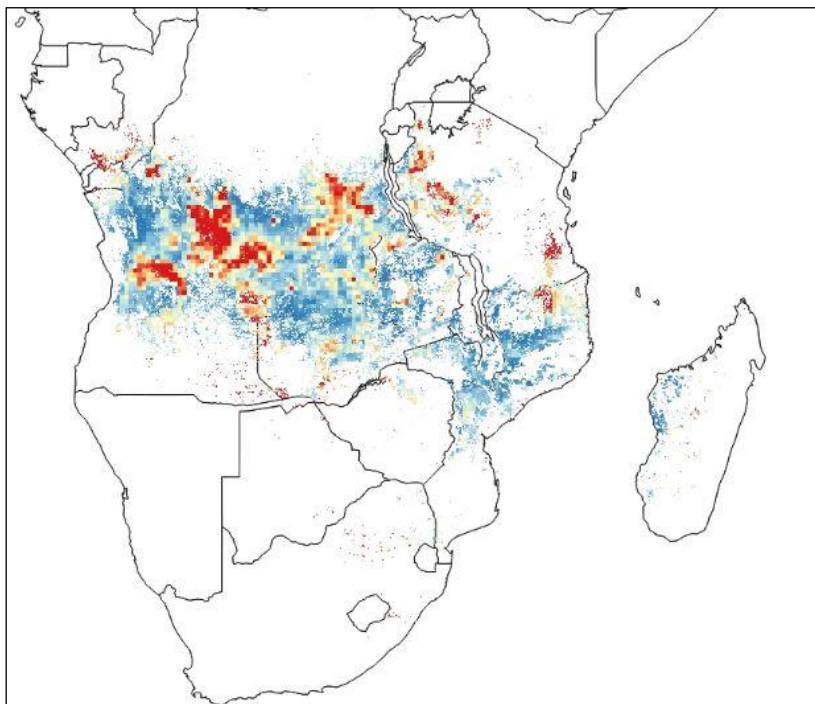


Figure 6.10. Detail of Figure 6.9 for southern hemisphere Africa.

7 Formatting subsystem verification

The formatting subsystem is used to generate the PSD compliant products from the burned area data produced using the BA processing subsystem.

7.1 Code sanity checks

As it is the last step of processing, and the actual user products are created here, lots of effort has been put into creating code sanity checks in the formatting subsystem. Note that these checks apply to the MERIS, the MODIS, the AVHRR, and the SFD production subsystem.

Most prominently, the actual data is checked several times in the processing of the grid product, because the aggregation happens in this step, which is the most error-prone part of the formatting:

- 1) If the computed area for any of the cells has negative size, there must have been a programming error somewhere else; the processing is stopped. Code:

```
private static void validate(double area, int index) {
    if (area < 0) {
        throw new IllegalStateException("area < 0 at target pixel "
+ index);
    }
}
```

- 2) If any cell has a positive error value, while it also has a zero BA value, the input data must be wrong, or there must have been a programming error; the processing is stopped. Code:

```
private static void validate(float[] errors, float[] ba) {
    for (int i = 0; i < errors.length; i++) {
        float error = errors[i];
        if (error > 0 && !(ba[i] > 0)) {
            throw new IllegalStateException("error > 0 && !(ba[i] >
0)");
        }
    }
}
```


	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
			Page	33

- 3) If the total burned area of a cell differs from the sum of burned area in each land cover class by more than 5%, the considered data must be wrong, or there must have been a programming error; the processing is stopped. Code:

```
private static void validate(float[] ba, List<float[]> baInLc) {
    int baCount = (int) IntStream.range(0, ba.length).mapToDouble(i -> ba[i]).filter(i -> i != 0).count();
    if (baCount < ba.length * 0.05) {
        // don't throw an error for too few observations
        return;
    }

    float baSum = (float) IntStream.range(0, ba.length).mapToDouble(i -> ba[i]).sum();
    float baInLcSum = 0;
    for (float[] floats : baInLc) {
        baInLcSum += IntStream.range(0, floats.length).mapToDouble(i -> floats[i]).sum();
    }
    if (Math.abs(baSum - baInLcSum) > baSum * 0.05) {
        CalvalusLogger.getLogger().warning("Math.abs(baSum - baInLcSum) > baSum * 0.05");
        CalvalusLogger.getLogger().warning("baSum = " + baSum);
        CalvalusLogger.getLogger().warning("baInLcSum = " + baInLcSum);
        throw new IllegalStateException("Math.abs(baSum - baInLcSum) > baSum * 0.05");
    }
}
```

- 4) When predicting the error, ensure that there are as many burned area pixels as area values; if this cannot be ensured, there must have been a programming error. Code:

```
float[] predictError(float[] burnedAreaInSquareMeters, double[] cellSizeInSquareMeters) throws ScriptException {
    if (burnedAreaInSquareMeters.length != cellSizeInSquareMeters.length) {
        throw new IllegalArgumentException("For each burned area pixel there must be exactly one cell size value.");
    }
    [...]
```

Also, there are a large number of easier checks that are also performed, for example the following:

- validate LC class; if not in Fire-LC-classes, fail
- validate input data: if not of correct size, fail
- validate "status" band of pre-processed data: if not existing or cannot be read, fail
- validate error prediction model: if it cannot be executed, fail
- validate error prediction process: if anything within process fails, fail whole formatting process
- validate input year: if not in 2002-2012 (MERIS) or 2015/2016 (SFD), fail
- validate geo-coding: if geo-coding cannot be applied, fail

7.2 Unit-level tests

There are a large amount of unit-level tests ensuring the correctness of the process. See the sections below for the respective report of running these tests, valid for the MERIS, the MODIS, and the SFD production.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		34

7.2.1 Generic tests

Test common utils:

```

-----
Test set: com.bc.calvalus.processing.fire.format.CommonUtilsTest
-----

Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 sec - in
com.bc.calvalus.processing.fire.format.CommonUtilsTest

```

Test the generic data source for grid product generation:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.AbstractFireGridData
SourceTest
-----

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.AbstractFireGridData
SourceTest

```

Test the generic mapper for grid product generation:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.AbstractGridMapperTe
st
-----

Tests run: 4, Failures: 0, Errors: 0, Skipped: 2, Time elapsed:
0.005 sec - in
com.bc.calvalus.processing.fire.format.grid.AbstractGridMapperTe
st

```

Test the area calculation code:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.AreaCalculatorTest
-----

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.474 sec - in
com.bc.calvalus.processing.fire.format.grid.AreaCalculatorTest

```

Test generic utility code for grid product generation:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.GridFormatUtilsTest
-----

Tests run: 3, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.137 sec - in
com.bc.calvalus.processing.fire.format.grid.GridFormatUtilsTest

```

7.2.2 MERIS-specific tests

Test the error prediction code:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.ErrorPredictorTest
-----

```

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		35

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
1.301 sec - in
com.bc.calvalus.processing.fire.format.grid.ErrorPredictorTest

Test the specific data source for MERIS data:

Test set:
com.bc.calvalus.processing.fire.format.grid.meris.MerisDataSource
eTest

Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.011 sec - in
com.bc.calvalus.processing.fire.format.grid.meris.MerisDataSource
eTest

Test the specific input format for MERIS grid formatting:

Test set:
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridInput
FormatTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0.001 sec - in
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridInput
FormatTest

Test the specific mapper for MERIS grid products:

Test set:
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridMappe
rTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridMappe
rTest

Test the specific reducer for MERIS grid products:

Test set:
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridReduc
erTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.meris.MerisGridReduc
erTest

Test the specific factory for MERIS NetCDF files:

Test set:
com.bc.calvalus.processing.fire.format.grid.meris.MerisNcFileFac
toryTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.001 sec - in
com.bc.calvalus.processing.fire.format.grid.meris.MerisNcFileFac
toryTest

Test the specific input format for MERIS pixel product:

 fire cci	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4		
		Issue	2.4	Date	11/01/2019
		Page			36

Test set:
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelInputFormatTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelInputFormatTest

Test the specific MergeMapper for MERIS pixel products:

Test set:
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelMergeMapperTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelMergeMapperTest

Test the specific reducer for MERIS pixel products:

Test set:
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelReducerTest

Tests run: 10, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.pixel.meris.MerisPixelReducerTest

7.2.3 SFD-specific tests

Test the specific data source for SFD data:

Test set:
com.bc.calvalus.processing.fire.format.grid.s2.S2FireGridDataSourceTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.s2.S2FireGridDataSourceTest

Test the specific input format for SFD grid formatting:

Test set:
com.bc.calvalus.processing.fire.format.grid.s2.S2GridInputFormatTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.s2.S2GridInputFormatTest

Test the aggregator which produces SFD pixel products:

Test set:
com.bc.calvalus.processing.fire.format.pixel.s2.JDAggregatorTest

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		37

```

-----
Tests run: 18, Failures: 0, Errors: 0, Skipped: 0, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.pixel.s2.JDAggregatorTest

```

Test the Mapper which finalises the SFD pixel products:

```

Test set:
com.bc.calvalus.processing.fire.format.pixel.s2.S2FinaliseMapper
Test

```

```

-----
Tests run: 8, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
59.506 sec - in
com.bc.calvalus.processing.fire.format.pixel.s2.S2FinaliseMapper
Test

```

Test the SFD aggregation strategy:

```

Test set: com.bc.calvalus.processing.fire.format.S2StrategyTest

```

```

-----
Tests run: 2, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0 sec - in com.bc.calvalus.processing.fire.format.S2StrategyTest

```

7.2.4 MODIS-specific tests

Test the MODIS data source for grid products:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.modis.ModisFireGridD
ataSourceTest

```

```

-----
Tests run: 8, Failures: 0, Errors: 0, Skipped: 2, Time elapsed:
0.545 sec - in
com.bc.calvalus.processing.fire.format.grid.modis.ModisFireGridD
ataSourceTest

```

Test the input format for MODIS grid formatting:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.modis.ModisGridInput
FormatTest

```

```

-----
Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0 sec - in
com.bc.calvalus.processing.fire.format.grid.modis.ModisGridInput
FormatTest

```

Test the specific MODIS grid mapper:

```

Test set:
com.bc.calvalus.processing.fire.format.grid.modis.ModisGridMappe
rTest

```

```

-----
Tests run: 2, Failures: 0, Errors: 0, Skipped: 1, Time elapsed:
0.913 sec - in
com.bc.calvalus.processing.fire.format.grid.modis.ModisGridMappe
rTest

```

Test the specific MODIS pixel aggregator:

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		38

Test set:
com.bc.calvalus.processing.fire.format.pixel.modis.ModisJDAggregatorTest

Tests run: 14, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec - in
com.bc.calvalus.processing.fire.format.pixel.modis.ModisJDAggregatorTest

7.2.5 AVHRR-specific tests

Test the data source for AVHRR grid formatting:

Test set:
com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrFireGridDataSourceTest

Tests run: 4, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 27.855 sec - in
com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrFireGridDataSourceTest

Test the input format for AVHRR grid formatting:

Test set: com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrGridInputFormatTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0 sec - in
com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrGridInputFormatTest

Test the specific AVHRR grid mapper:

Test set: com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrGridMapperTest

Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.007 sec - in
com.bc.calvalus.processing.fire.format.grid.avhrr.AvhrrGridMapperTest

Test the specific AVHRR aggregation mode:

Test set: com.bc.calvalus.processing.fire.format.grid.avhrr.CollocationOpTest

Tests run: 1, Failures: 0, Errors: 0, Skipped: 1, Time elapsed: 0 sec - in
com.bc.calvalus.processing.fire.format.grid.avhrr.CollocationOpTest

7.3 Calvalus monitoring

7.3.1 FireCCI41

There are two different Calvalus workflows for the formatting: one controls the processing of the grid product, the other controls the processing of the pixel product.

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
			Page	39

For the grid product, the expected number of output files is $2 * 7 * 12$ (two files a month, 2005-2011) = 168 files. This is what we find in the report:

```
thomas ~/fire-inst (FIRE)$ cat grid-format.status
168 created, 0 running, 0 backlog, 168 processed, 0 failed
```

Also, looking into the result file system shows the same amount of files:

```
thomas@feeder01:/calvalus/projects/fire/meris-PSD$ find -name
"*nc" | wc -l
168
```

For the pixel product, the expected number of output files is $6 * 12 * 7$ (6 zones each month, 2005-2011) = 504 files. This is what we find in the report:

```
thomas ~/fire-inst (FIRE)$ cat pixel-format.status
504 created, 0 running, 0 backlog, 504 processed, 0 failed
```

Also, looking into the result file system shows the same amount of files:

```
thomas@feeder01:/calvalus/projects/fire/meris-PSD-pixel$ find -
name "*AREA*.tar.gz" | wc -l
504
```

In order to check the file sizes, it has been ensured that – apart from the control files created by Calvalus – there are no zero-sized result files:

```
thomas@feeder01:/calvalus/projects/fire/meris-PSD $ find . -size
0 | grep -v _SUCCESS | grep -v part-r-00000 | wc -l
0
thomas@feeder01:/calvalus/projects/fire/meris-PSD-pixel $ find .
-size 0 | grep -v _SUCCESS | grep -v part-m-00000 | grep -v
part-r-00000 | wc -l
0
```

7.3.2 FireCCISFD11

Similarly to the MERIS case, there are two kinds of Calvalus workflow for the formatting: one for the processing of the grid product, the other for processing of the pixel product. For the pixel product, the expected number of outputs is $116 * 12$ (116 5-degree-areas, 12 months) = 1392 files. This is what the report states:

```
thomas ~/fire-inst (FIRE)$ cat s2-pixel.status
1392 created, 0 running, 0 backlog, 1392 processed, 0 failed
```

For the grid product, the expected number of outputs is simply 12: one for each month of 2016. This is the output of the processing report:

```
thomas ~/fire-inst (FIRE)$ cat s2-grid.status
12 created, 0 running, 0 backlog, 12 processed, 0 failed
```

7.3.3 FireCCI50/1

As for the other formatting cycles, there are two different Calvalus workflows for the formatting: one controls the processing of the grid product, the other controls the processing of the pixel product.

 fire cci	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
			Page	40

For the grid product, using the example of FireCCI51, the expected number of output files is $17 * 12 + 1$ (one file a month, Dec 2000 - Dec 2017) = 205 files. This is what we find in the report:

```
thomas ~/fire-inst (FIRE)$ cat modis-grid.status
205 created, 0 running, 0 backlog, 206 processed, 0 failed
```

Also, looking into the result file system shows the same amount of files:

```
thomas@feeder01:/calvalus/projects/fire/modis-grid$ find -name
"*nc" | wc -l
205
```

For the pixel product, the expected number of output files is $6 * 17 * 12 + 6$ (6 zones each month, Dec 2000- Dec 2017) = 1230 files. This is what we find in the report:

```
thomas ~/fire-inst (FIRE)$ cat modis-pixel.status
1230 created, 0 running, 0 backlog, 1230 processed, 0 failed
```

Also, looking into the result file system shows the same amount of files:

```
thomas@feeder01:/calvalus/projects/fire/modis-pixel $ hdfs dfs -
find `pwd` -name "*fv5.1-JD*" | wc -l
1230
```

In order to check the file sizes, it has been ensured that – apart from the control files created by Calvalus – there are no zero-sized result files:

```
thomas@feeder01:/calvalus/projects/fire/modis-grid $ find . -
size 0 | grep -v _SUCCESS | grep -v part-r-00000 | wc -l
0

thomas@feeder01:/calvalus/projects/fire/modis-pixel $ find . -
size 0 | grep -v _SUCCESS | grep -v part-m-00000 | grep -v part-
r-00000 | wc -l
0
```

7.3.4 FireCCILT10

There is only a single Calvalus workflow for the formatting, which controls the processing of the grid product.

For this product, the expected number of output files is $36 * 12 - 8$ (Jan 1982 - Dec 2017, May-Dec 1994 missing) = 424 files. This is what we find in the report:

```
thomas ~/fire-inst (FIRE)$ cat avhrr-grid.status
432 created, 0 running, 0 backlog, 424 processed, 8 failed
```

So all files for which there are input data have been successfully processed.

7.4 Visual inspection

The PSD-compliant products have been thoroughly verified by the algorithm developers and the project manager of the project, to assure that all the layers of the products comply with the PSD specifications. Some examples of the visual inspection are shown below. The products were evaluated by the revisers, a report was sent to the system engineer, he performed the corrections, and then the revisers evaluated the outputs and updated the report. This process was done iteratively until all layers complied with the specifications.

7.4.1 FireCCI41

See below for sample images of the first half of June, 2008, globally (Figure 7.1), and January 2011, Africa (Figure 7.2); both images are configured to show only BA with a slightly transparent world map in the background, with – arbitrarily chosen – different colours indicating different days of burn. Apart from these sample images confidence level, distribution of Land Cover classes and – in the case of the grid product – number of patches and observed area fraction have also been inspected. See Figure 7.3 to Figure 7.10 for examples.

Date and location were chosen by the algorithm developers, based on previously gathered experience. The images have been compared to reference data kept by the algorithm developers, and have been confirmed as showing the expected results. The full list of tiles and years is provided in Annex 2.

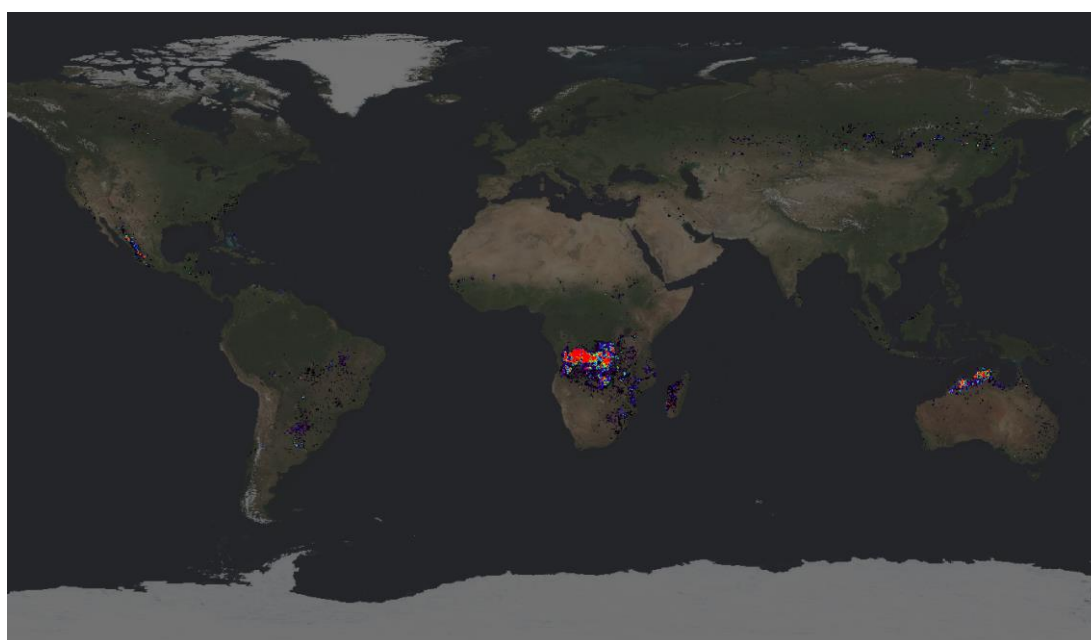


Figure 7.1: PSD compliant global BA product for June 2008.

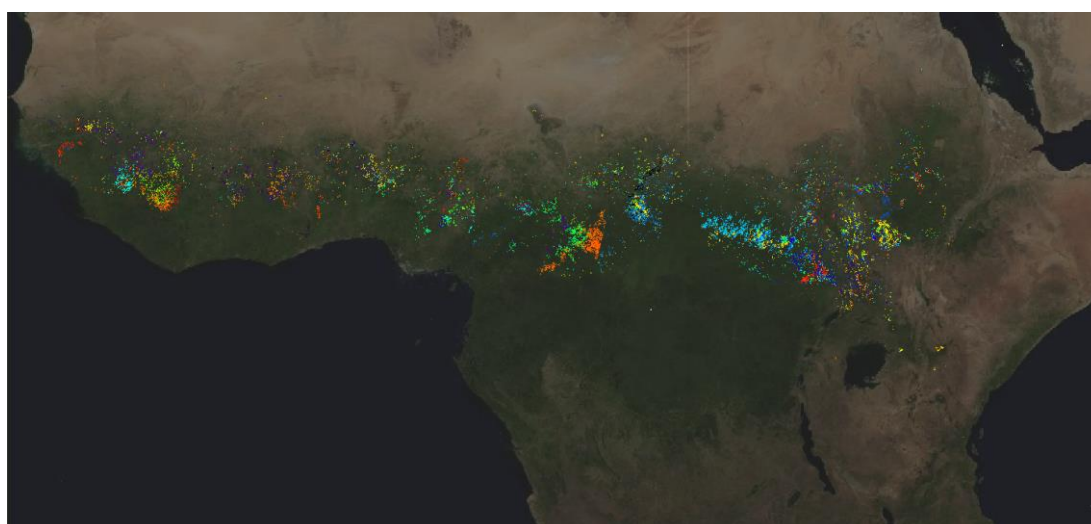


Figure 7.2: PSD compliant BA product for January 2011, Africa.

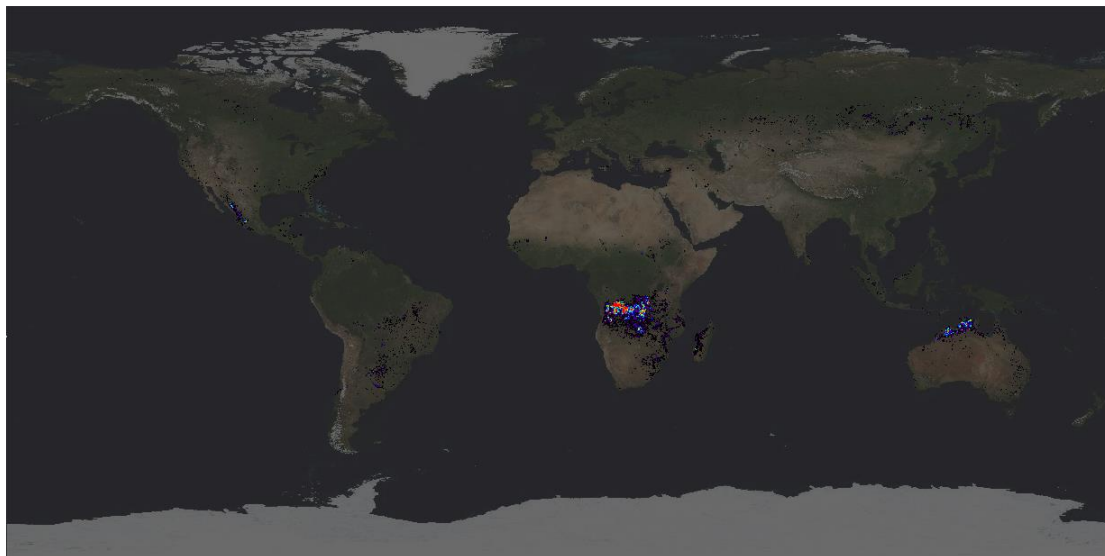


Figure 7.3: PSD compliant global standard error for June 2008.

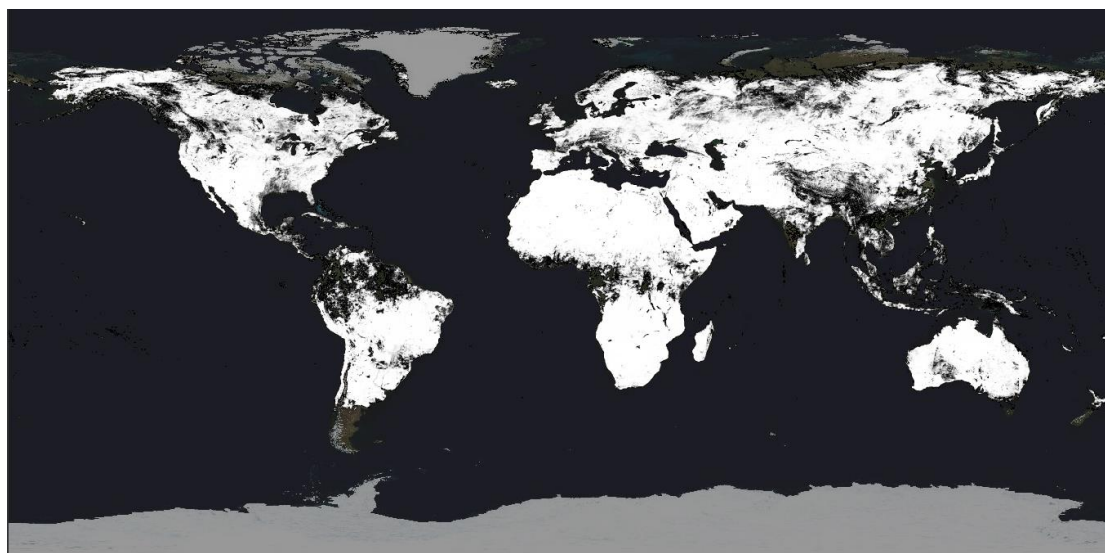


Figure 7.4: PSD compliant global fraction of observed area for June 2008.

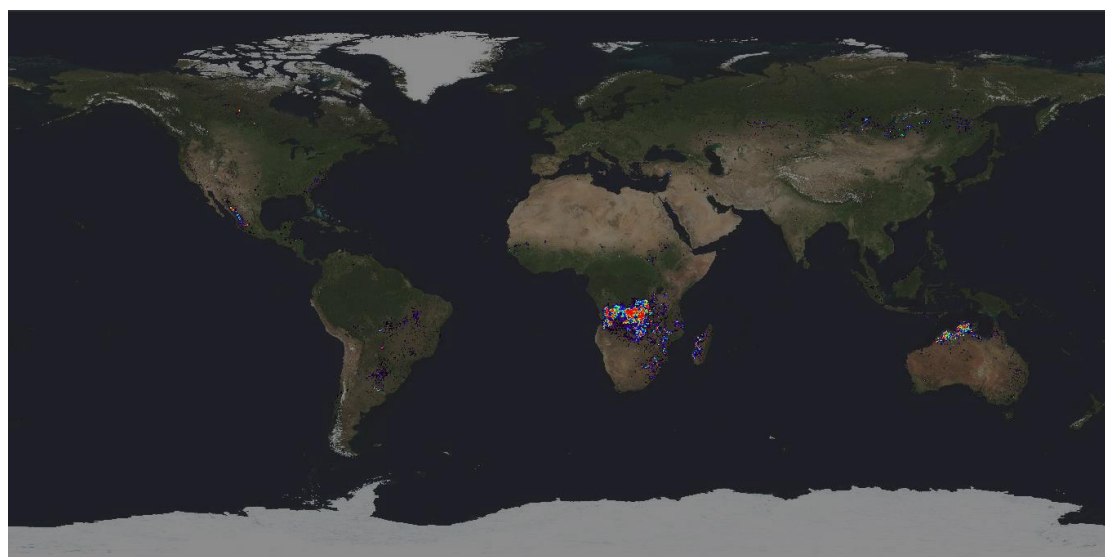


Figure 7.5: PSD compliant global number of patches for June 2008

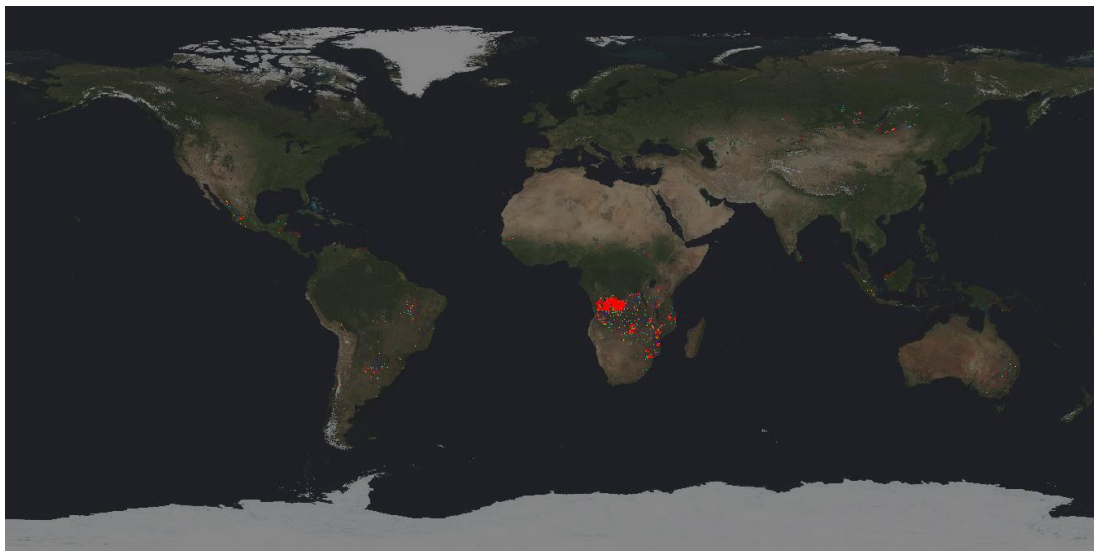


Figure 7.6: PSD compliant BA in LC class 10 (cropland) for June 2008.

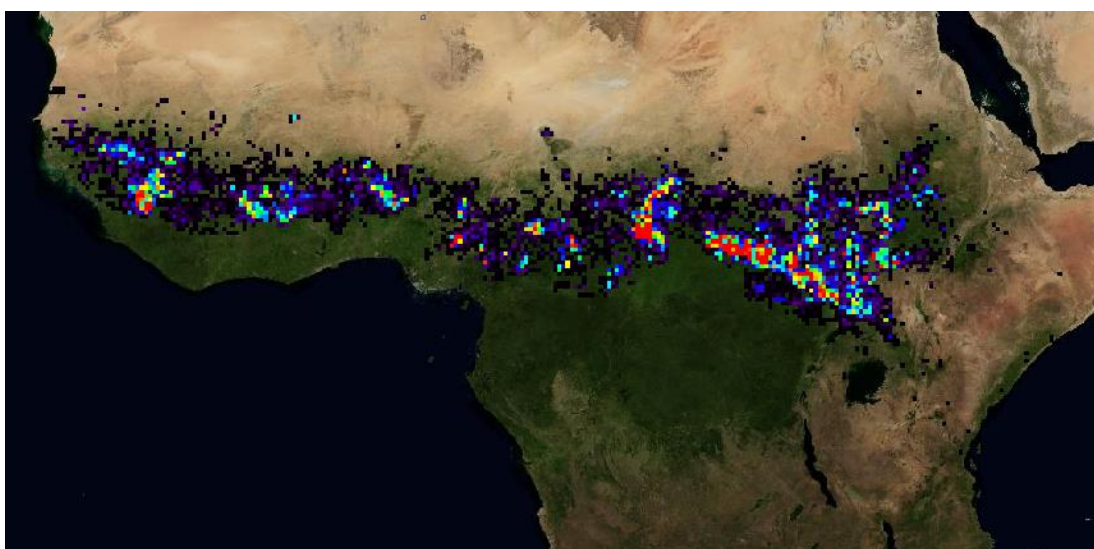


Figure 7.7: PSD compliant standard error product for January 2011, Africa.

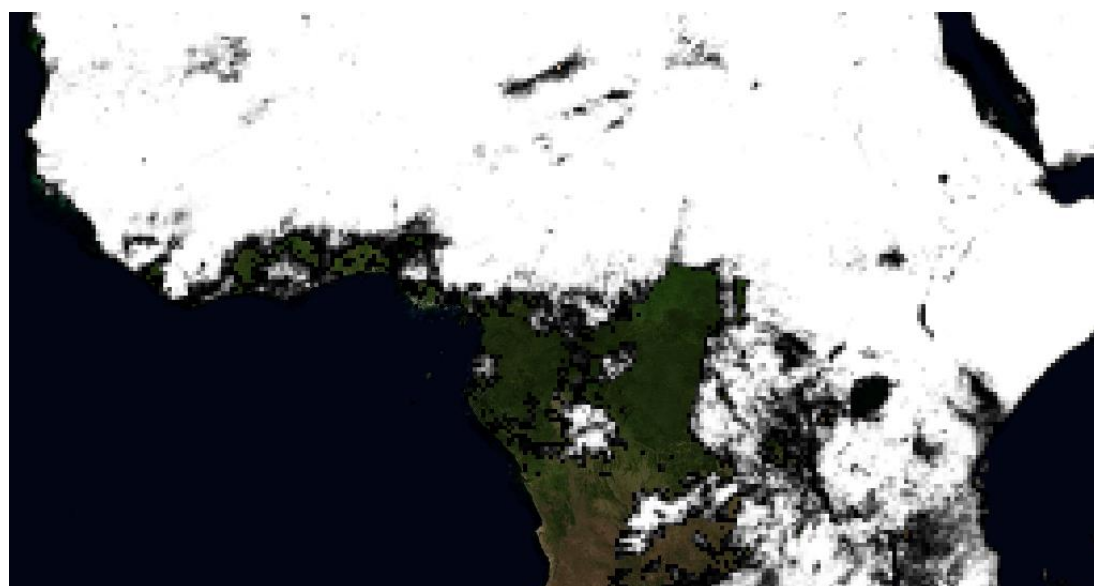


Figure 7.8: PSD compliant fraction of observed area for January 2011, Africa.

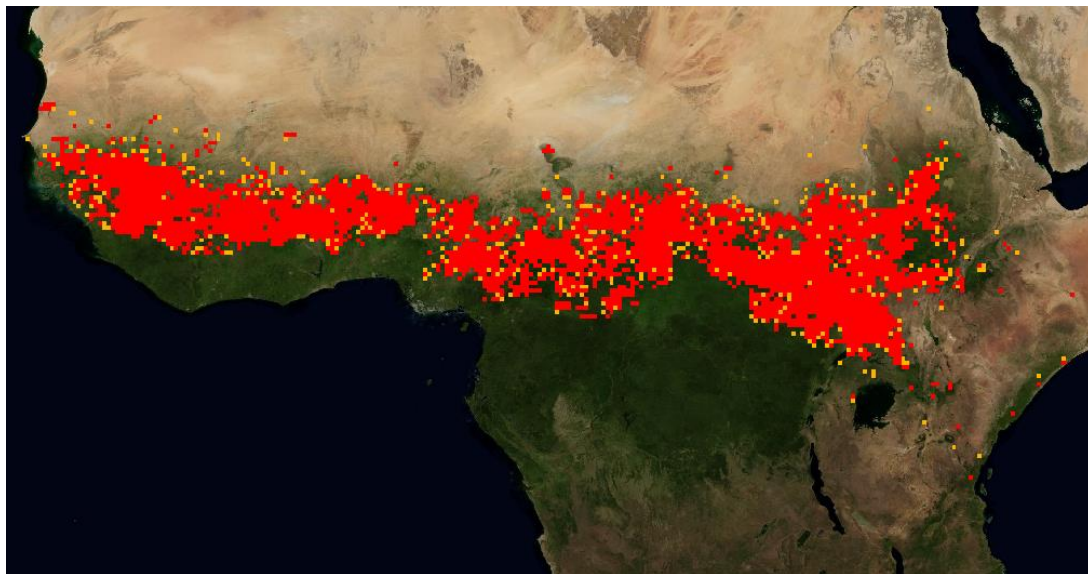


Figure 7.9: PSD compliant number of patches for January 2011, Africa.

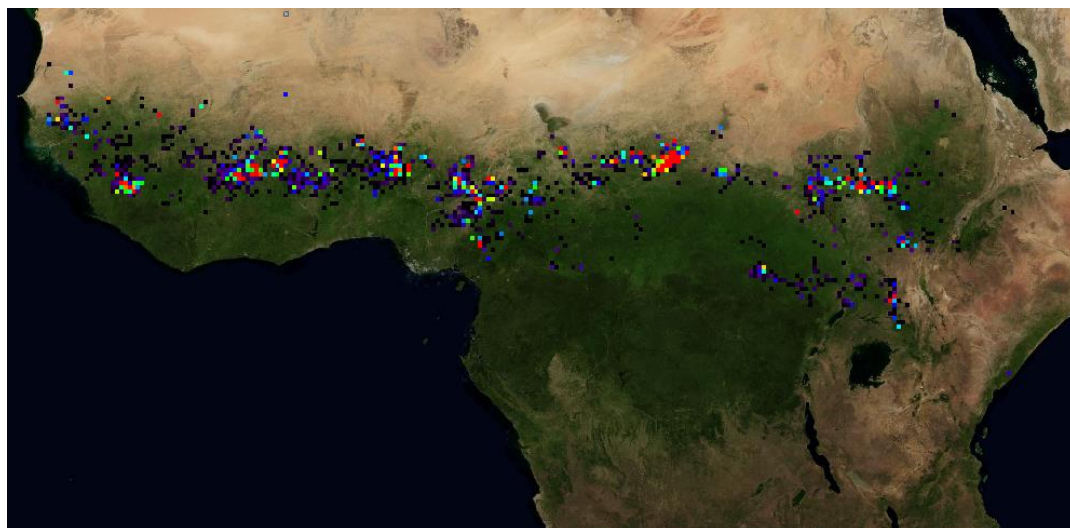


Figure 7.10: PSD compliant BA in LC class 10 (cropland) for January 2011, Africa

7.4.2 FireCCISFD11

Visual inspection of the SFD products were done for multiple pixel products. See Figure 7.11 for a sample image of January 2016, pixel product tile h36v16, configured to show only BA with a slightly transparent world map in the background, with – arbitrarily chosen – different colours indicating different days of burn. Apart from these sample images, confidence level and distribution of Land Cover classes have also been inspected.

Date and location were chosen by the algorithm developers, based on previously gathered experience. The images have been compared to reference data kept by the algorithm developers, and have been confirmed as showing the expected results.

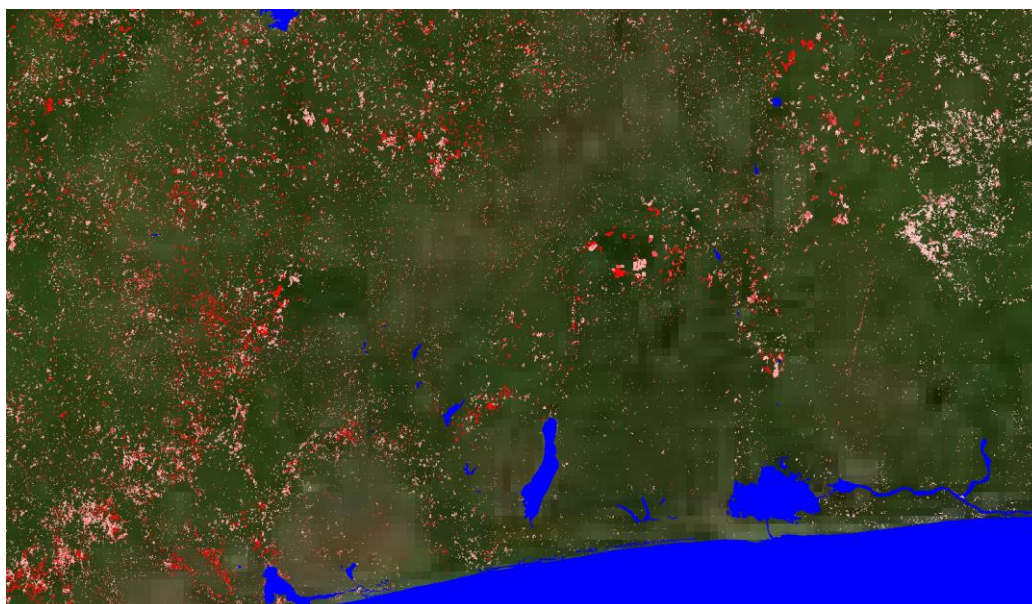


Figure 7.11: Pixel product, Jan 2016, tile h36v16

Multiple artefacts have been identified using the visual inspection (see Figure 7.12 for an example). All these artefacts could be tracked to the Sen2Cor 2.2.3 pre-processing, which introduced these kinds of errors, and could not be fixed in the processing cycle. These errors were informed to the Sen2Cor developers, but for technical reasons (performance, processing time, and schedule) it was not possible to wait until a fixed version was developed to process FireCCISFD11.

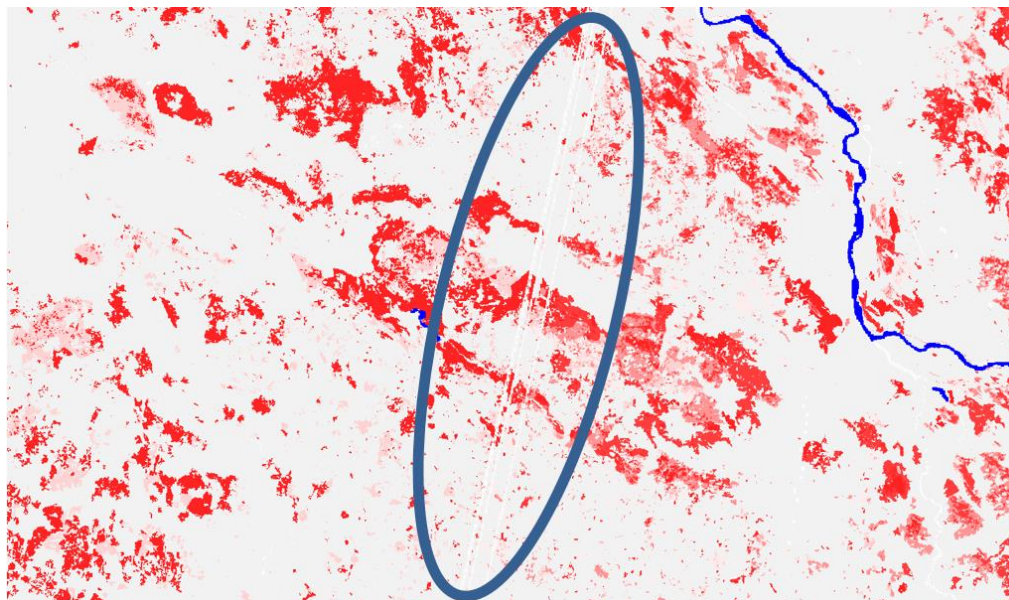


Figure 7.12: Pixel product with artefact

7.4.3 FireCCI50/1

Similarly to the MERIS dataset, there has also been validation done by visual inspection and comparison with the BA data, apart from consistency checks performed through python scripts by the algorithm developers. This has been done for a random selection of pixel products in each zone, and for all grid products. See below for sample images of the grid product of December, 2004, globally (Figure 7.13 to Figure 7.18).

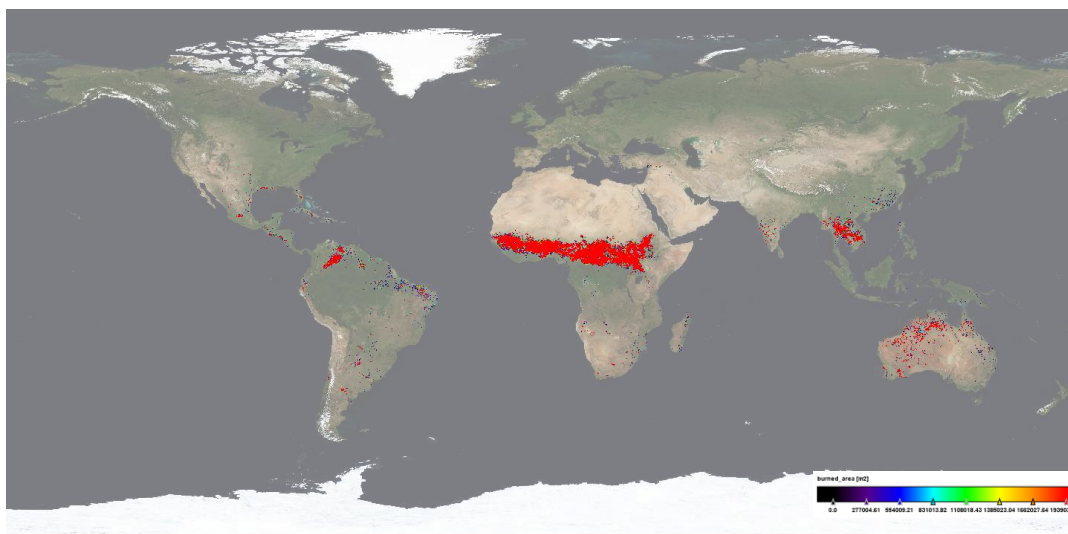


Figure 7.13: PSD compliant global BA product of Dec 2004

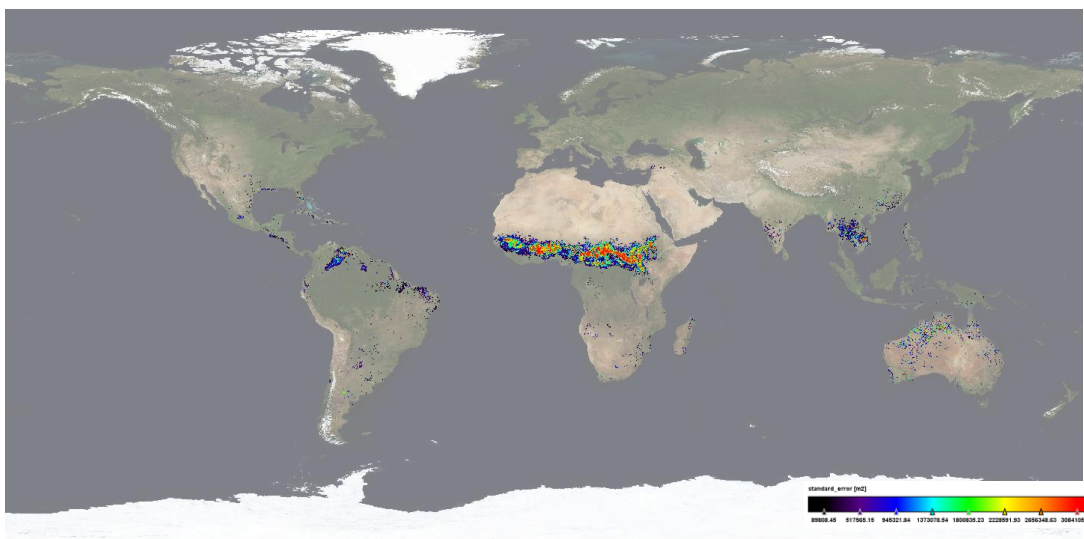


Figure 7.14: PSD compliant global standard error of Dec 2004

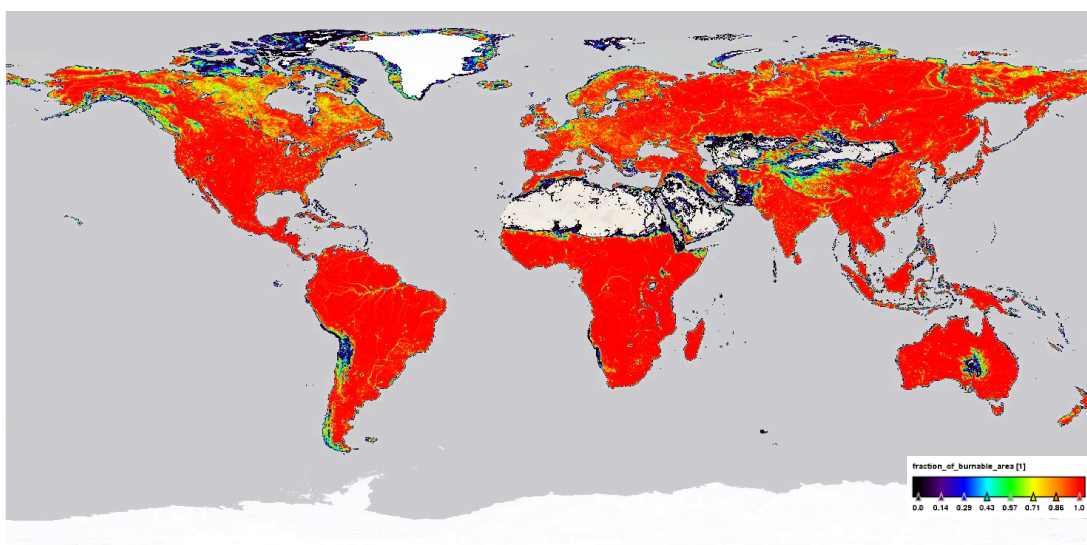


Figure 7.15: PSD compliant fraction of burnable area of Dec 2004

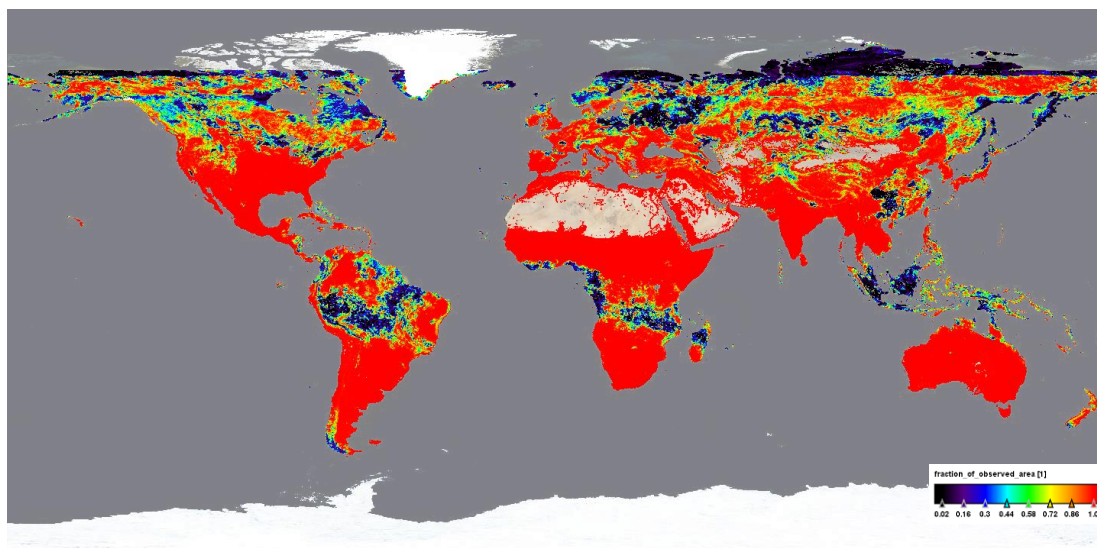


Figure 7.16: PSD compliant global fraction of observed area of Dec 2004

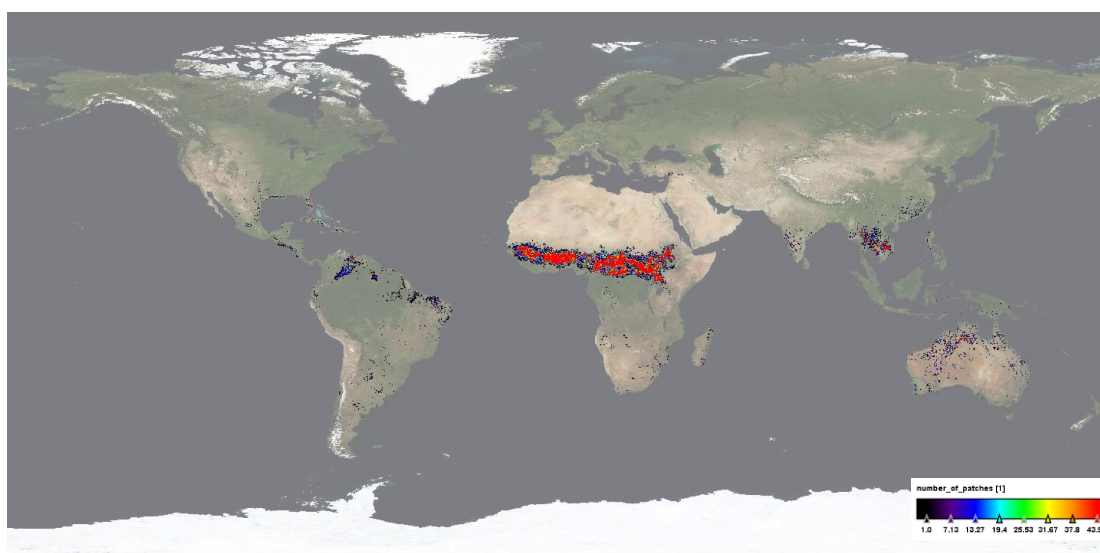


Figure 7.17: PSD compliant global number of patches of Dec 2004

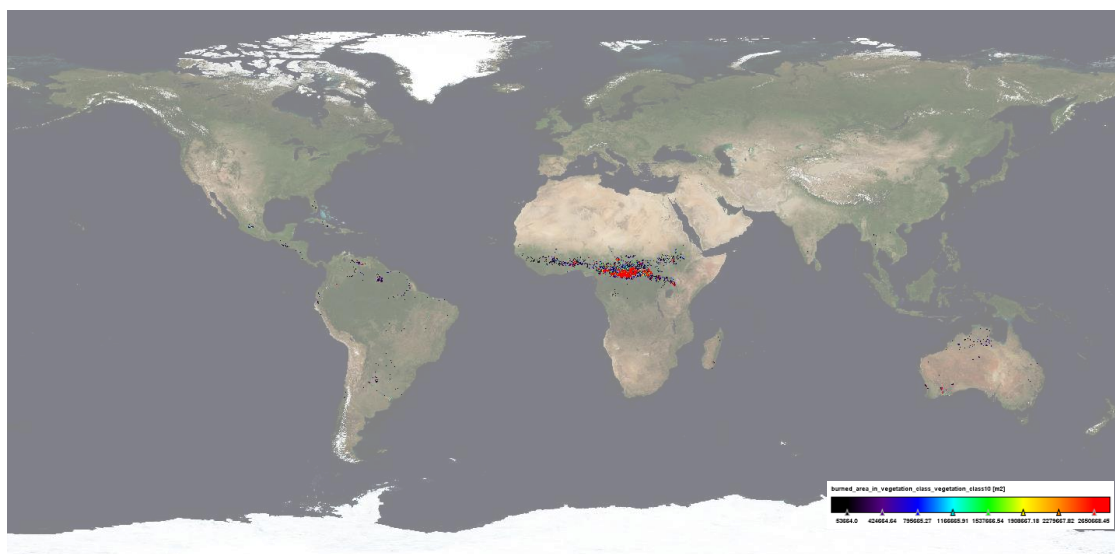


Figure 7.18: PSD compliant BA in LC class 10 (cropland) for Dec 2004

Figures Figure 7.19 to Figure 7.22 show the values for a sample pixel product (Africa in March 2006) corresponding to FireCCI50, which has been used for validation with the non-aggregated data the BA algorithm produced.

All data have been confirmed as showing the expected results.

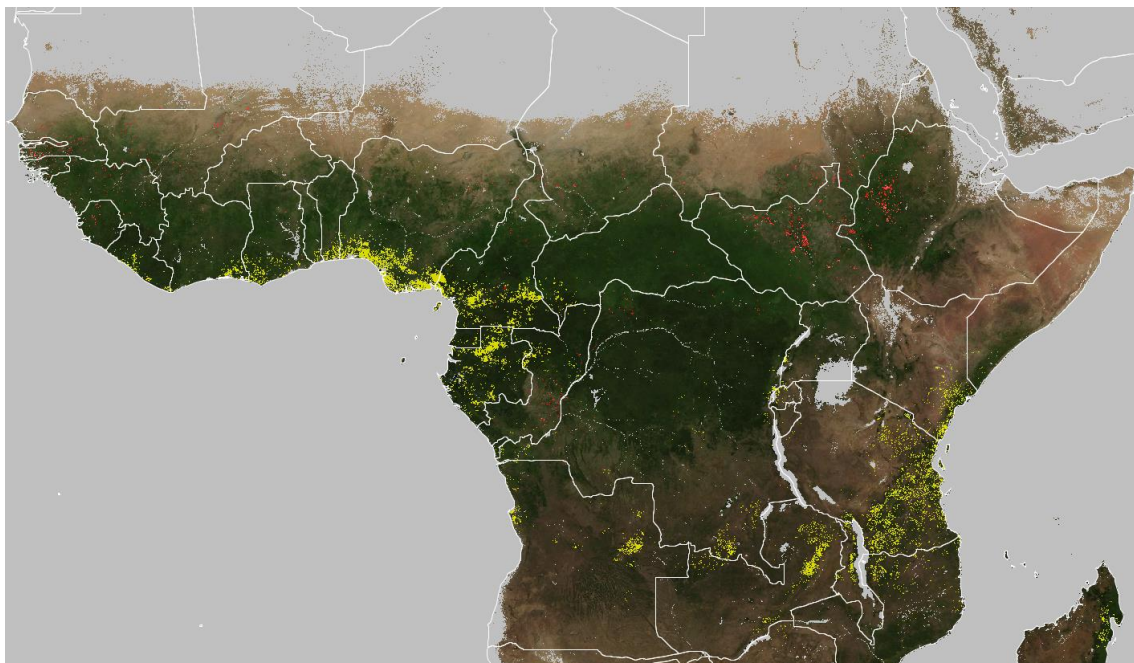


Figure 7.19: PSD compliant JD pixel product for March 2006, Africa. Grey pixels indicate unburnable areas, yellow pixels indicate unobserved areas, and red pixels indicate burned areas.

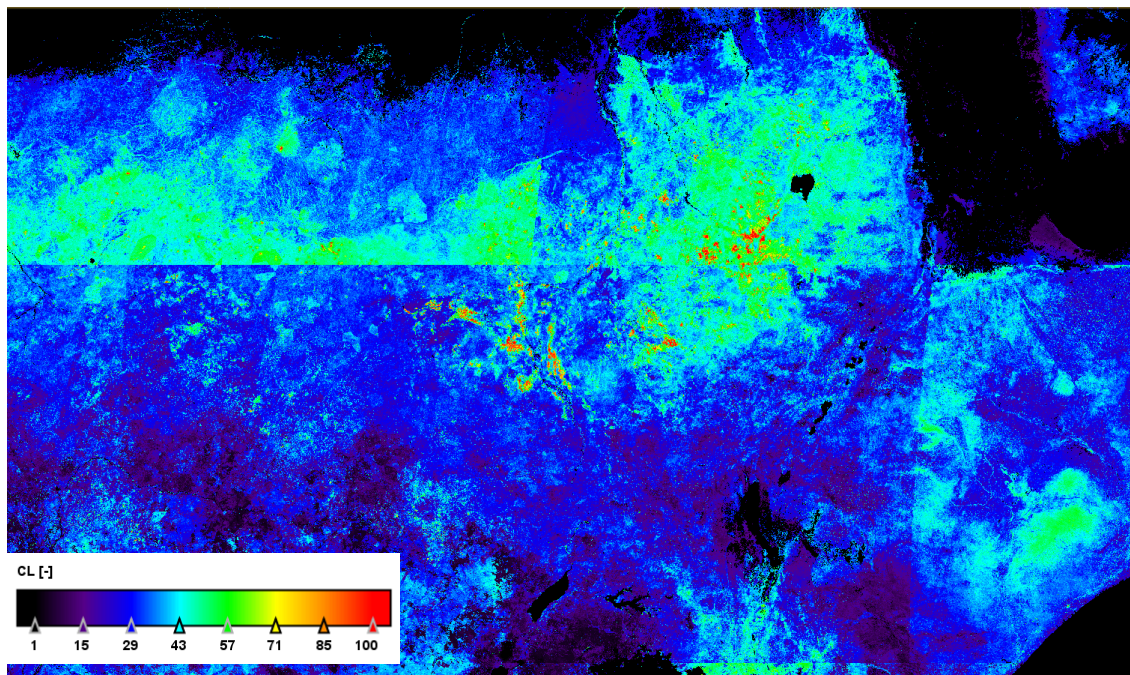


Figure 7.20: PSD compliant CL pixel product for March 2006, Africa for FireCCI50. It is clearly visible that different MODIS input tiles show different CL values. Still, the information correlates with the burned areas visible in the upper right of Figure 7.19.

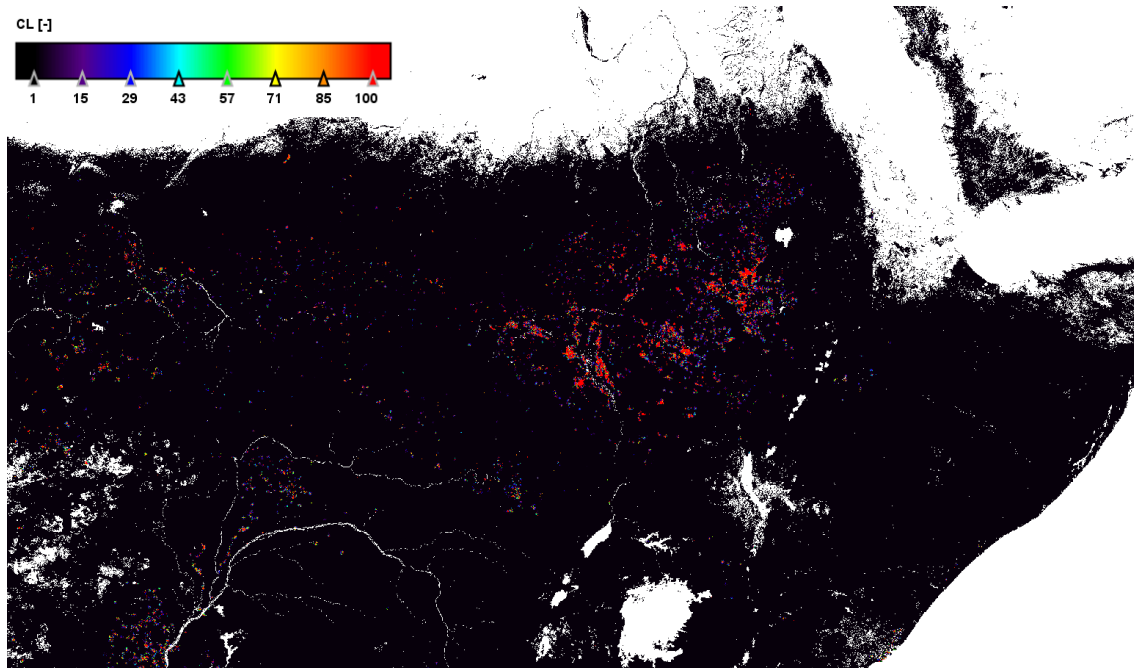


Figure 7.21: PSD compliant CL pixel product for March 2006, Africa for FireCCI51. Although the colour scaling is identical, the values have changed, because the method for confidence level calculation has been improved. Also, the border effects are not visible anymore.

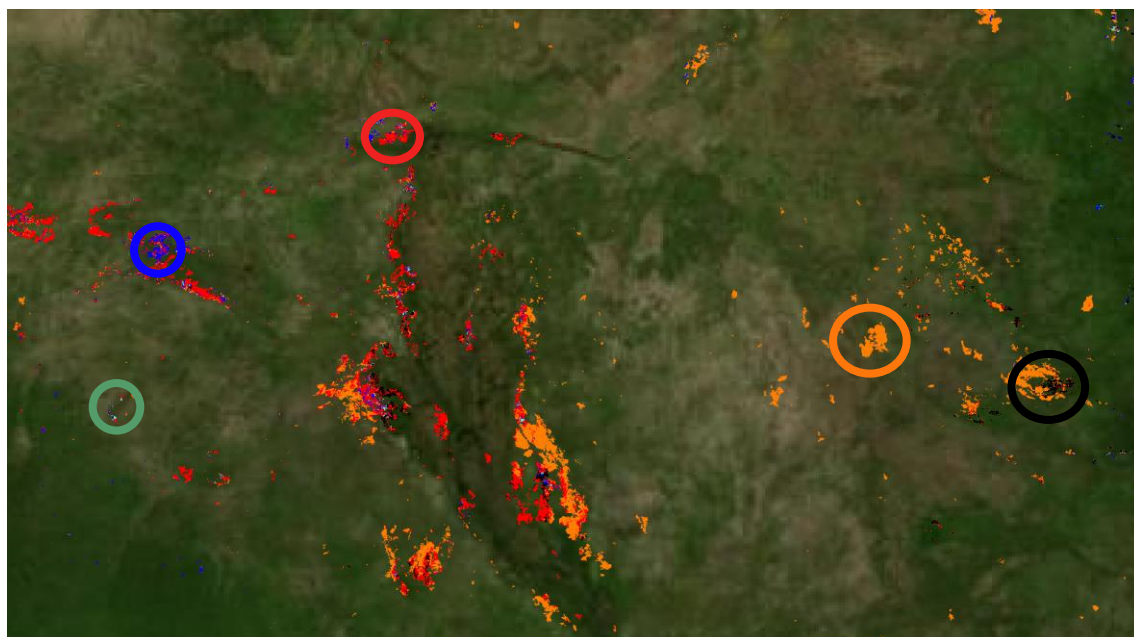


Figure 7.22: Detail of PSD compliant LC pixel product for March 2006, Africa. Black pixels indicate LC class 10, turquoise pixels indicate LC class 30, blue pixels indicate LC class 60, orange pixels indicate LC class 120, red colour indicates LC class 180. Occurrences of the respective pixels have been marked by respectively coloured circles for easier visibility.

7.4.4 FireCCILT10

For validation of the LTDR FireCCILT10 dataset, February 2000 has been (arbitrarily) chosen. See below for sample images of the different layers of that product.

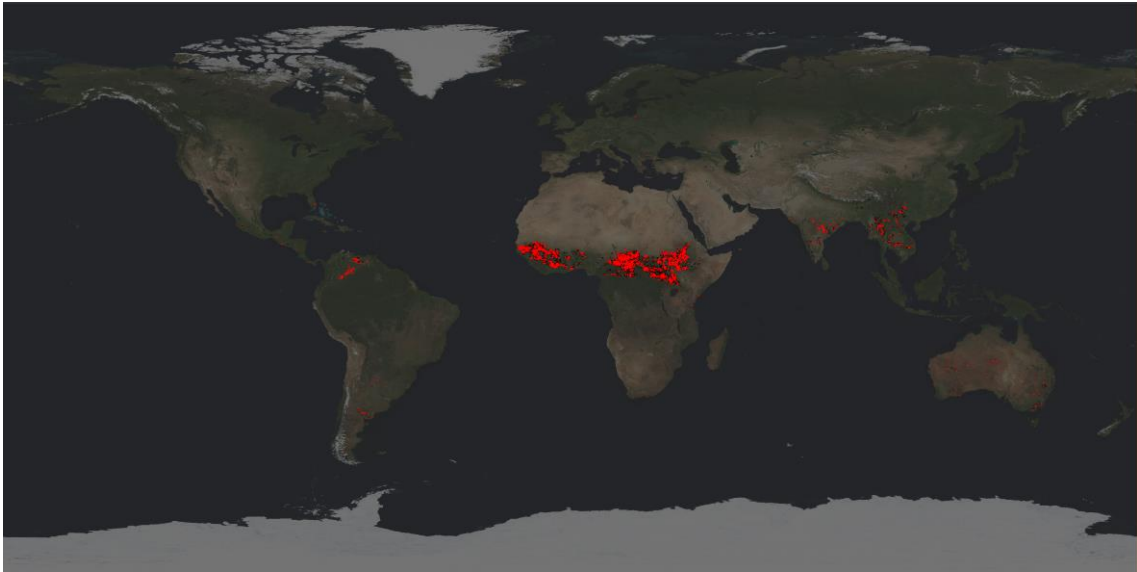


Figure 7.23: Burned Area, global, February 2000

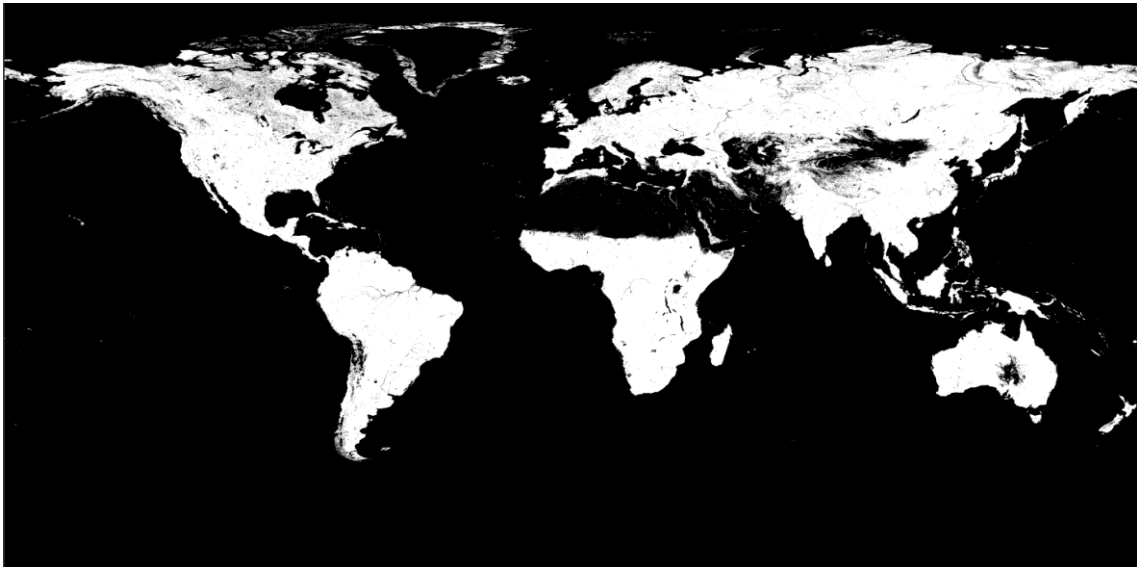


Figure 7.24: Fraction of burnable area, February 2000

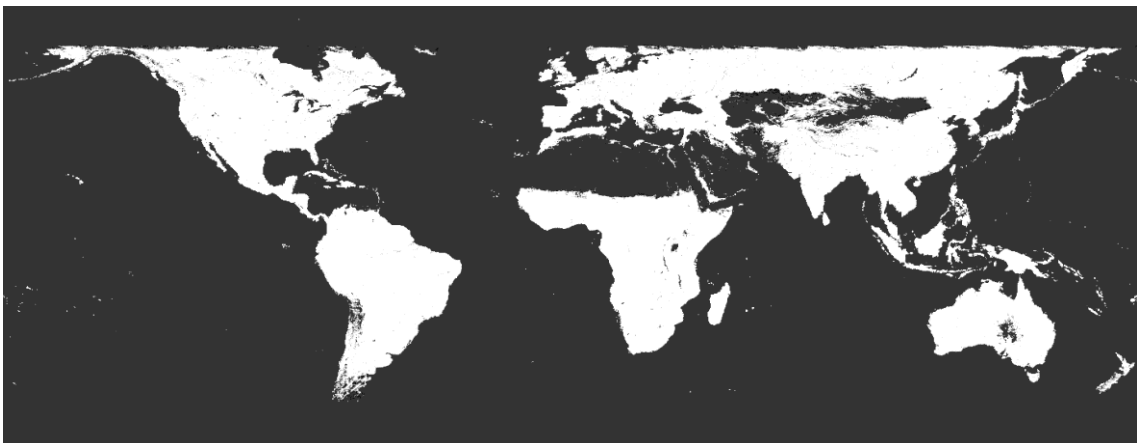


Figure 7.25: Fraction of observed area, February 2000

	Fire_cci System Verification Report	Doc.	Ref.: Fire_cci_D3.2_SVR_v2.4	
		Issue	2.4	Date 11/01/2019
		Page		51

Annex 1: Acronyms and Abbreviations

AD	Applicable Document
BA	Burned Area
CCI	Climate Change Initiative
CNFDB	Canadian National Fire Database
DSWG	Data Standards Working Group
ESA	European Space Agency
FireCCI41	MERIS Fire_cci v4.1
FireCCI50	MODIS Fire_cci v5.0
FireCCI51	MODIS Fire_cci v5.1
FireCCILT10	AVHRR-LTDR Fire_cci v1.0
FireCCISFD11	Sentinel-2 SFD Fire_cci v1.1
FRAP	Fire and Resource Assessment Program
LC	Land Cover
MERIS	Medium Resolution Imaging Spectrometer
MODIS	Moderate Resolution Imaging Spectroradiometer
NAFI	North Australian Fire Information
PSD	Product Specification Document
SDR	Surface Directional Reflectance
SR	Surface Reflectance
SFD	Small Fire Database
SSD	System Specification Document
SVR	System Verification Report

Annex 2: List of reference tiles and dates used for visual inspection of FireCCI41 products

a) Visual inspection of MERIS BA outputs

Tile	Year
v03h07	2007
v03h07	2008
v03h23	2008
v03h24	2008
v03h29	2008
v03h30	2008
v04h20	2008
v05h08	2008
v06h25	2008
v07h16	2008
v07h17	2008
v07h18	2008
v07h19	2008
v08h10	2008
v08h11	2008
v08h16	2008
v08h17	2008
v08h18	2008
v08h19	2008
v08h20	2008
v08h21	2008
v09h20	2008
v10h20	2008
v08h20	2011

b) Visual inspection of FireCCI41 pixel product

File	Year	Confirmed compliant by UAH's developer
Zone 1 (North America)	2007	yes
Zone 1 (North America)	2008	yes
Zone 2 (South America)	2008	yes
Zone 3 (Europe)	2008	yes
Zone 4 (Asia)	2008	yes
Zone 5 (Africa)	2008	yes
Zone 5 (Africa)	2011	yes

c) Visual inspection of FireCCI41 grid product

File	Confirmed compliant by UAH's developer
20080507-ESACCI-L4_FIRE-BA-MERIS-fv03.1.nc	yes
20080607-ESACCI-L4_FIRE-BA-MERIS-fv03.1.nc	yes
20080107-ESACCI-L4_FIRE-BA-MERIS-fv03.1.nc	yes