POLITECNICO
MILANO 1863

SCHOOL OF CIVIL, ENVIRONMENTAL AND
LAND MANAGEMENT ENGINEERING

BURNED AREA DETECTOR

Development of a QGIS plugin for mapping burned areas from Sentinel-2 images

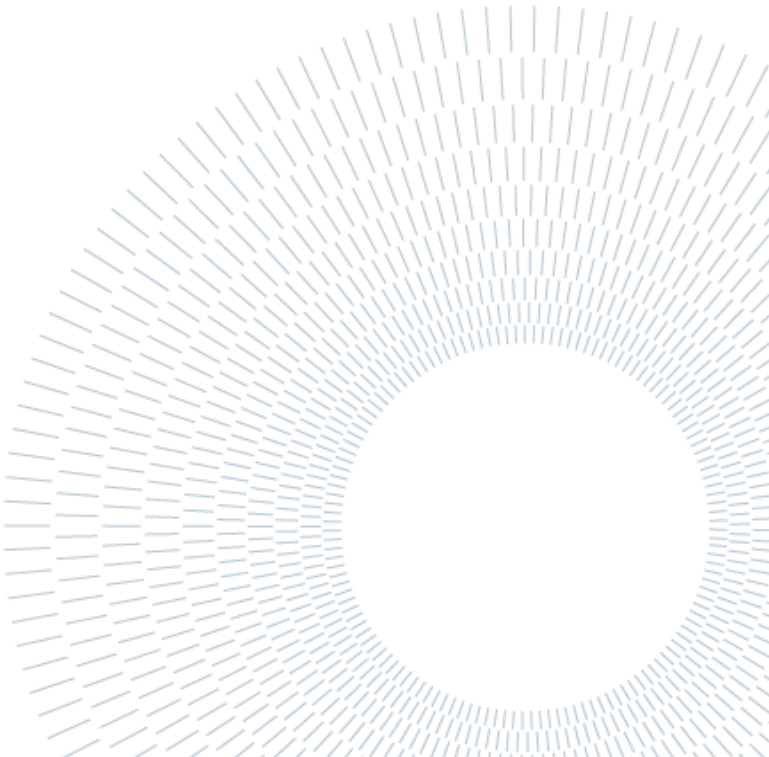
GEOINFORMATICS ENGINEERING

MASTER DEGREE THESIS

Author: **Thomas Martinoli**

Student ID: 952907
Advisor: Prof. Giovanna Sona
Co-advisor: Dr. Daniela Stroppiana

Academic Year: 2021-2022



Abstract

The purpose of the work was the development of a plugin for Burned Area (BA) Detection from satellite multi-spectral images (Sentinel-2 [1]). The plugin implements a burned area [2] and a burn severity [3] mapping algorithms to produce output geo-products depicting the fire affected areas and the level of damage induced to the vegetation.
The approach for burned area identification was based on soft computing techniques and on the integration of features (spectral bands and indices) derived from Sentinel-2 multispectral satellite images; integration is done by implementing ordered weighted average operators (OWAs) from the fuzzy set theory. A Region Growing (RG) contextual algorithm is implemented as the final step to balance omission and commission errors.
Instead, the severity computation exploits the spectral bands (NIR and SWIR) of the S-2 multispectral satellite images and, in particular, the difference of the Normalized Burn Ratio (dNBR) index. The dNBR between pre-fire and post-fire S-2 images was computed and exploited for pixel level classification into degrees of burn severity.
The plugin (BAD: Burned Area Detector) was developed within a GIS environment and can be downloaded for free from the public repository in GitHub (BAD_QGISplugin [4]) and installed in QGIS.
The choice was intended to make the tool more usable by users with different programming backgrounds and knowledge.
The plugin was designed and developed to be a flexible tool that leaves each user free to choose and use her/his inputs, to set the parameters and to choose the features to be integrated to map burned areas.
Moreover, it was designed with modular tabs to run single steps of the algorithm and/or the entire process through its intuitive user interface (UI). The output products are: Burned Area Map and the Severity Map but also a combination of the two: Severity Map for identified burned pixels.
The BAD plugin was tested on real scenarios with S-2 images for mapping burned areas affected by a wildfire event: Portugal (2017), Spain (2022), Chile (2023). The BA and burn severity maps produced by BAD were validated by comparison with reference datasets produced by the Copernicus Emergency Management Service (EMS). The validation showed that the output BA maps are very accurate with ranges for the accuracy metrics of [2.4% : 9.24%] for omission error, [1.84% : 15.66%] for commission error, [88.81% : 96.57%] for Dice Coefficient and [-6.78% : 8.04%] for Relative bias.
The performance of the BAD plugin was also tested by measuring processing time as a function of the input image size and the number of seed pixels.

These results showed how the computational time depended on the selected parameters; the step requiring more processing time was the RG algorithm.
The BAD plugin proved to be a valuable tool for mapping burned areas and burn severity through a flexible and user friendly interface; the availability within a GIS environment allows the user to exploit it synergically with QGIS functionalities for the analysis and the process of geospatial data and products.

Key-words: Multi-spectral images, Sentinel-2, Burned Area Mapping, Burn Severity Mapping, Fuzzy theory, Region Growing algorithm, GIS, plugin development

Abstract in italiano

Lo scopo del lavoro è stato lo sviluppo di un plugin per il rilevamento di aree bruciate ("Burned Area", BA) da immagini satellitari multispettrali (Sentinel-2 [1]).

Il plugin implementa un algoritmo di mappatura delle aree bruciate [2] e un algoritmo di mappatura della gravità ("Severity") delle aree soggette a incendio [3] per produrre geoprodotti che descrivono le aree colpite dall'evento e il livello di danno prodotto dal fuoco alla vegetazione.
L'approccio per l'identificazione delle aree bruciate si è basato su tecniche di soft computing e sull'integrazione di "Features" (bande spettrali e indici) derivate da immagini satellitari multispettrali Sentinel-2; l'integrazione viene effettuata utilizzando operatori di aggregazione "Ordered Weight Avarage" (OWA) dalla teoria degli insiemi fuzzy. Un algoritmo contestuale di Region Growing (RG), che sfrutta come layer di semi e di growing diversi OWA, viene implementato come fase finale per bilanciare gli errori di omissione e commissione.
Il calcolo della gravità sfrutta invece le bande spettrali (NIR e SWIR) delle immagini satellitari multispettrali S-2 e, in particolare, la differenza multi-temporale dell'indice Normalized Burn Ratio (dNBR). Il dNBR, calcolato tra le immagini S-2 pre-incendio e post-incendio, è stato utilizzato per la classificazione dei gradi di gravità dei danni per ciascun pixel.
Il plugin (BAD: Burned Area Detector) è stato sviluppato all'interno di un ambiente GIS e può essere scaricato gratuitamente dalla cartella pubblica su GitHub (BAD_QGISplugin [4]) e installato in QGIS.
La scelta è stata fatta per rendere lo strumento più fruibile da utenti con conoscenze e background di programmazione diversi.
Il plugin è stato progettato e sviluppato per essere uno strumento flessibile che lascia ogni utente libero di scegliere e utilizzare diversi input, di impostare i parametri e di scegliere le "Features" da integrare per mappare le aree bruciate. Inoltre, è stato progettato con schede modulari ("TAB") per eseguire i singoli passaggi dell'algoritmo e/o l'intero processo attraverso la sua intuitiva interfaccia ("User Interface").
I prodotti di output sono: Mappa dell'area bruciata e Mappa della gravità, ma anche una combinazione delle due: Mappa della gravità per i pixel identificati come bruciati.
Il plugin BAD è stato testato su scenari reali di eventi di incendio per la mappatura delle aree bruciate da immagini S-2 selezionate per ogni evento prima e dopo il passaggio del fuoco: Portogallo (2017), Spagna (2022), Cile (2023).
Le mappe di BA e di "severity" delle aree colpite da incendio prodotte da BAD sono

state validate mediante confronto con i dati di riferimento forniti dal Servizio di gestione delle emergenze (EMS) di Copernicus. La validazione ha mostrato che le mappe di BA prodotte sono molto accurate, con intervalli per le metriche di accuratezza di [2,4% : 9,24%] per l'errore di omissione, [1.84% : 15.66%] per l'errore di commissione, [88.81% : 96.57%] per il coefficiente Dice e [-6.78% : 8.04%] per il bias relativo.
Oltre alla stima dell'accuratezza dei prodotti del plugin BAD, è stato stimato anche il tempo di calcolo fattore particolarmente rilevante nel caso di applicazioni di monitoraggio in tempo (quasi) reale. I risultati hanno mostrato come il tempo di processing dipenda dalle dimensioni dell'immagine di input e dal numero di semi iniziali nell'algoritmo di Region Growing. Il RG è risultato essere la fase che richiede più tempo di elaborazione e di produzione delle mappe di area bruciata.
Il plugin BAD si è rivelato un valido strumento per la mappatura delle aree bruciate e della gravità provocata dall'incendio attraverso un'interfaccia flessibile e di facile utilizzo; la disponibilità all'interno di un ambiente GIS consente all'utente di sfruttarlo sinergicamente con le funzionalità di QGIS per l'analisi e l'elaborazione di dati e prodotti geospaziali.

Parole chiave: Immagini multispettrali, Sentinel-2, Mappatura di Aree Bruciate, Mappatura della Gravità (Severity) delle Aree Bruciate, Teoria Fuzzy, Algoritmo Region Growing, GIS, Sviluppo di plugin

Ringraziamenti

Ci tengo a ringraziare la Prof.ssa Giovanna Sona per aver supervisionato la realizzazione di questo lavoro e per avermi introdotto a una nuova realtà: il CNR-IREA di Milano.
Qui ho avuto modo di entrare in contatto con persone appassionate, professionali e cortesi, che mi hanno sostenuto dalle fasi di sviluppo sino alla completa realizzazione di questo lavoro. In particolare ringrazio la Dott.ssa Daniela Stroppiana e il Dr. Matteo Sali per la loro costante presenza, sempre disponibili per chiarimenti e confronti.

Vorrei, infine, ringraziare i miei cari e tutti gli amici che mi hanno accompagnato in questo percorso. Sono grato alla mia famiglia per avermi sostenuto, supportato, ma soprattutto sopportato, fin dal primo giorno.

Contents

1 Introduction

This thesis focuses on the phenomena of wildfires and the implementation of an algorithm to map the area affected by fires from remote sensing data. Goal of the work is to develop a new tool that allows each user to better quantify the areas affected by fires (perimeters) from satellite images and hence quantify the phenomena of wildfires and their impacts.

According to Copernicus [5], wildfires, also known as wildland fires or forest fires, develop where there is available combustible vegetation (fuel) and an ignition source (heat), such as from lightning strikes or from human activity.
Hence, in order to start and maintain fires, three elements are essential (Figure 1.1):

- fuel: combustible material, such as oils, liquids, textiles, paper, gasses, wood, rubber, plastics, etc. In the studied case, generally, vegetation fuel is considered.
  The fuel's moisture content, shape, size and quantity can influence how easily and at what temperature the fuel burns. For instance, when fuel has low moisture content, fire starts more easily and spreads more rapidly.
  Therefore, fuel's characteristics are a fundamental parameter in order to estimate the fire intensity;
- heat: a heat source (natural or artificial) is responsible for the initial ignition of the phenomenon. Heat is also needed to maintain the fire and enable it to spread, in fact it dries out and preheats nearby fuel and warms surrounding air;
- oxygen: most fires require at least 16% oxygen to burn, and since oxygen makes up about 21% of the Earth's atmosphere, optimal conditions for the combustion are met in most of the cases.

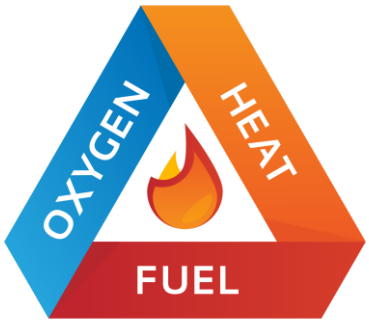


Figure 1.1 Fire triangle

If even one of these components is missing or is removed, fire cannot start or spread. Furthermore, there are others factors that can influence the fire development phenomena:

- topography and wind affect the direction and the speed of spread;
- the season, the temperature and relative humidity influence the wildfires phenomena behaviour, for example: in summer, like happens in the Mediterranean region, the air temperature is high and the humidity low, these conditions generate the best scenario for wildfire onset and propagation. This aspect can be observed in Figure 1.2: from June to September the total burnt area increases more rapidly than in the other months. Moreover, looking at the graph (Figure 1.2), in 2021 the amount of area affected by wildfire is higher than the average for the reference period (2008-2020). Behaviour due to the persistent high temperatures and drought conditions in some Mediterranean regions, as described in the Copernicus' section "Mediterranean summer extremes" [6].

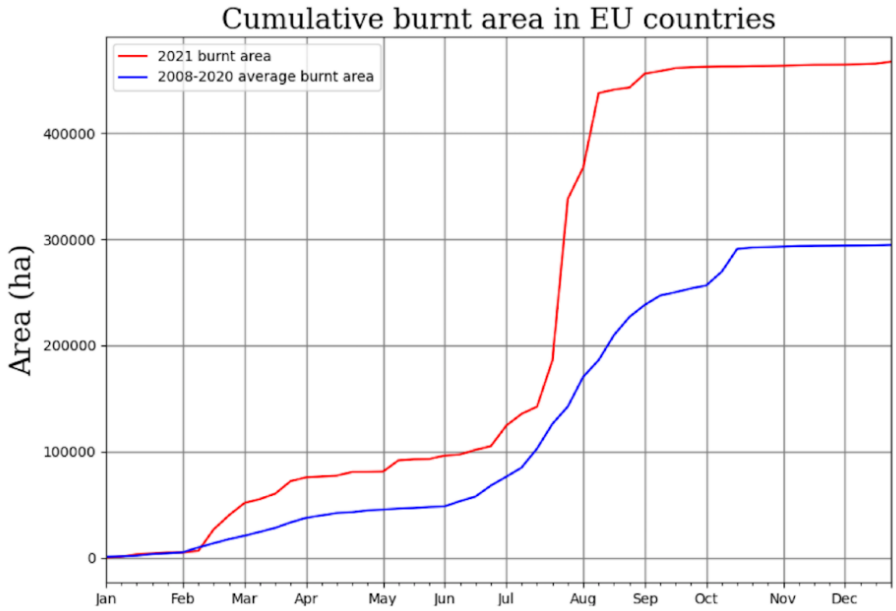


Figure 1.2 Cumulative burnt areas in 2021 in European Union (EU) countries (red) and average for the reference period 2008–2020 (blue). Data source: EFFIS. Credit: EFFIS/Copernicus EMS.

As reported in the article "Focus on changing fire regimes: interactions with climate, ecosystems, and society" [7] fires are a natural, inevitable and fundamental part of many ecosystems. Indeed, they act to:

- regulating atmospheric O2;
- driving plant evolution;
- determining the distribution of biomes and plant communities.

However, according to the article published by NASA: "Six trends to know about fire season in the western U.S" [8], there has been a steady increase in the number of fires. In particular the article focuses on the western U.S. (Figure 1.3)

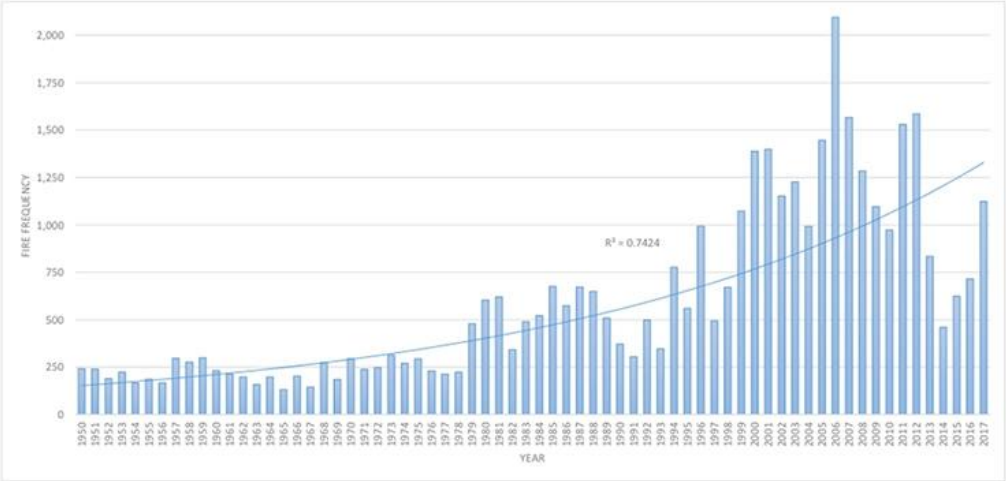


Figure 1.3 Frequency of wildfire from 1950 to 2017 in western U.S

Furthermore, another section of the article shows how those fires are also burning more hectares of land: the number of megafires (wildfires that burn more than 40,500 hectares) has increased in the past two decades (Figure 1.4).

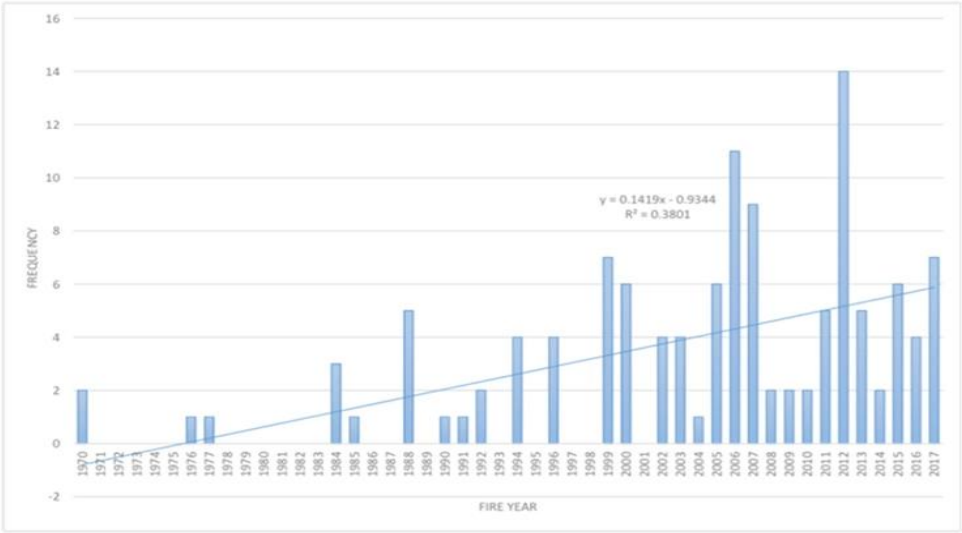


Figure 1.4 Frequency of megafire from 1970 to 2017 in western U.S

Therefore, a fire that spreads over a wider area, with a longer duration and is located near population centers has more dangerous and relevant consequences.

Consequences due to the fire destruction and the products generated during the combustion, like: gases (carbon dioxide, methane, carbon monoxide, etc.), particulates and ashes.
These substances, in the wrong concentration, are polluting agents that can significantly affect the water, the air and the soil quality.
Therefore multiple can be the effects on an ecosystem caused by a wildfire, as reported in the article "The Environmental Impact of Wildfires" [9] and by WHO (World Health Organization [10]), the main are:

- the soil's ability to absorb water is significantly compromised, a problem in case of flood events;
- the loss of vegetation can significantly alter an ecosystem by increasing erosion, reducing nutrient availability in the soil, and posing a heightened risk for disease and pest infestations;
- watersheds may retain higher levels of nitrogen and dissolved carbon dioxide for 15 years after a wildfire, reducing drinking water quality within surrounding communities, also wildlife may be affected;
- moreover the victims and injuries, also human health can be compromised by the substances released into the air:
    - eye, nose, throat and lung irritation;
    - decreased lung function, including coughing and wheezing;
    - pulmonary inflammation, bronchitis, exacerbations of asthma, and other lung diseases;
    - exacerbation of cardiovascular diseases, such as heart failure.

For this reason it is essential to study, analyze and understand this phenomenon as much as possible.

Data about this phenomenon can be firstly collected by applying Remote Sensing (RS) techniques, and successively processed in Geographic Information System (GIS) environment in order to supply products ready to be analyzed and interpreted like:

- Active Fire Map;
- Burned Area Map;
- Severity Map.

1.1. Remote Sensing

Remote Sensing (RS) is the discipline dealing with the process of detecting and monitoring the physical characteristics and conditions of an object on the Earth surface by measuring its reflected and/or emitted radiation at a distance (typically from satellite or aircraft), without touching the object itself. The radiation reflected and/or emitted from the Earth surface is measured by the sensor to extract information on the type of surface and its characteristics and conditions.
In RS the first distinction is based on the characteristics of the sensors involved:

- active sensors;
- passive sensors.

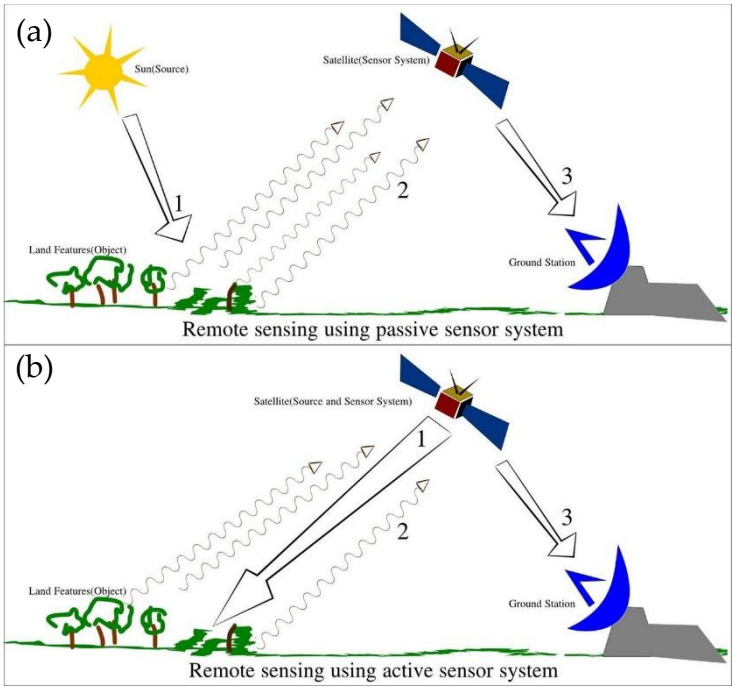


Figure 1.5 Passive RS (a); active RS (b)

In active RS the incoming radiation is produced by the instrument itself and the sensor measures the proportion of the radiation that is reflected or backscattered from the target.

Instead, in the case of passive multi-spectral sensors, the information is collected on the basis of the interaction of the Earth's surface with the Electromagnetic (EM) radiation coming from the Sun, that is the source of illumination.
So, passive sensors measure the radiation emitted and/or reflected by a surface or surrounding areas.

Hence, in the wavelength range where only the reflection occurs, these sensors can take measurements only during the day when the Sun is illuminating the Earth's surface. During the night only radiation emitted by the surface can be measured in the wavelength range of the thermal infrared, this is the radiation that is naturally emitted by the Earth as a function of the surface temperature.
Furthermore, depending on the satellite characteristics and the features of the sensor installed on board it is possible to have different levels of resolution. The are four different kinds of resolution when products from satellites are considered:

- the temporal resolution: it is the revisit frequency of the satellite to a particular location.
- the spatial resolution: it is the at-ground representation of an individual detector in a satellite sensor array. Therefore, the spatial resolution indicates the dimension of the area scanned by the satellite that will be represented later on in the processed image as a pixel.
- the spectral resolution: it is the ability of the instrument to distinguish features in the electromagnetic spectrum;
- the radiometric resolution: it is a determination of the incremental level of intensity or reflectance that can be represented or distinguished by the system. The higher the radiometric resolution, the more capable the device will be of detecting differences in intensity or reflectance. Generally this measure is represented with a bit number.

During this thesis work passive sensors are considered, therefore the data analyzed come from the interaction between Earth's surface and EM radiation coming from the Sun.

Electromagnetic (EM) waves are formed when a variable electric field (red line in Figure 1.6) couples with a variable magnetic field (blue line in Figure 1.6).
Frequency (f) and wavelength ($\boldsymbol{\lambda}$) of the EM radiation are related through the following equation:

$$f=c\boldsymbol{\lambda}$$

Formula 1.1 relationship between frequency and wavelength of EM wave

where c is the speed of light (~ 3x10^8 m/s).

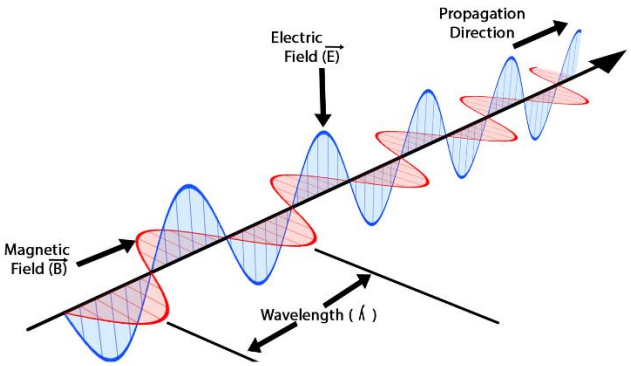


Figure 1.6 ElectroMagnetic wave, blue line represents the Electric field, red line represents Magnetic field

The relation implies that the shorter the wavelength, the higher the frequency.
For RS applications, wavelength (or frequency) is the most important characteristic of the EM radiation since it determines the type of interaction with the surface as well as the information that can be extracted from remote sensing data.

A useful representation of the range of frequencies (and wavelengths) is the EM spectrum (EMS). The "position" of an EM wave within the electromagnetic spectrum (Figure 1.7) can then be characterized by either its frequency of oscillation or its wavelength.

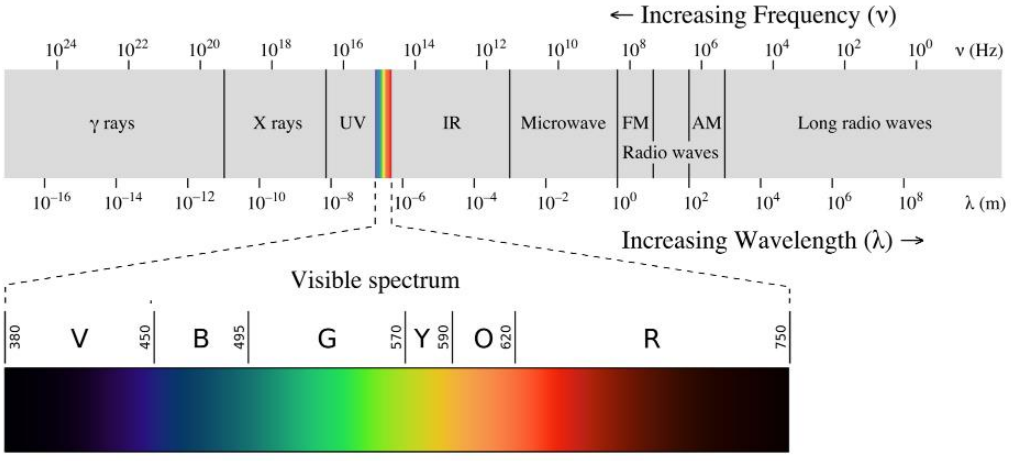


Figure 1.7 Electromagnetic spectrum regions and wavelengths

The EM spectrum is arbitrarily divided into regions and the most important ones for multi-spectral RS are the visible and infrared (from near infrared to middle infrared and thermal infrared wavelengths).

Before the incoming solar radiation reaches the Earth's surface it travels through Earth's atmosphere where particles and gases can be scattered and/or absorbed. These processes depend on the wavelength and gases absorb electromagnetic energy in very specific regions of the spectrum; in RS the wavelength ranges of the spectrum which are not influenced by atmospheric absorption are called atmospheric windows (Figure 1.8). The regions of the spectrum are therefore the most useful for remote sensing.

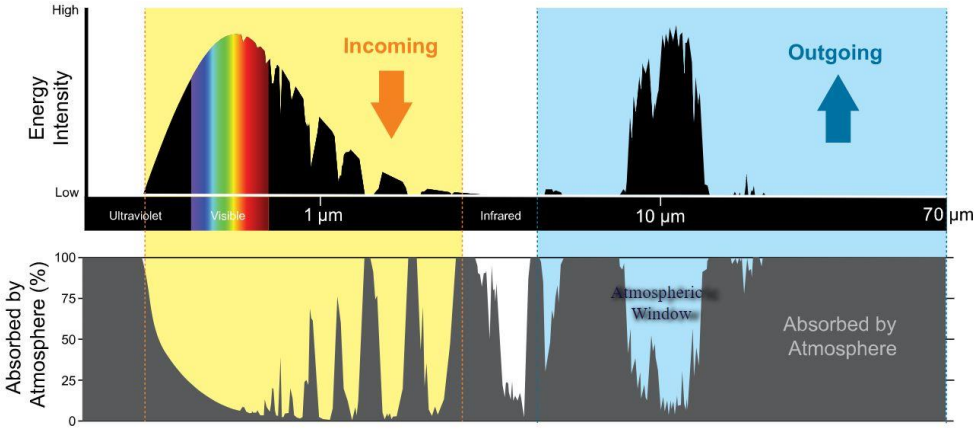


Figure 1.8 Incoming energy from the sun and outgoing energy from the earth relative to the electromagnetic spectrum.

The solar radiation that is not absorbed or scattered by the atmosphere can reach the Earth's surface where all objects interact with incoming solar radiation that can be reflected, absorbed, or transmitted in different proportions depending on the wavelength (Figure 1.7) and surface characteristics and conditions (spectral signature/spectral response Figure 1.9).

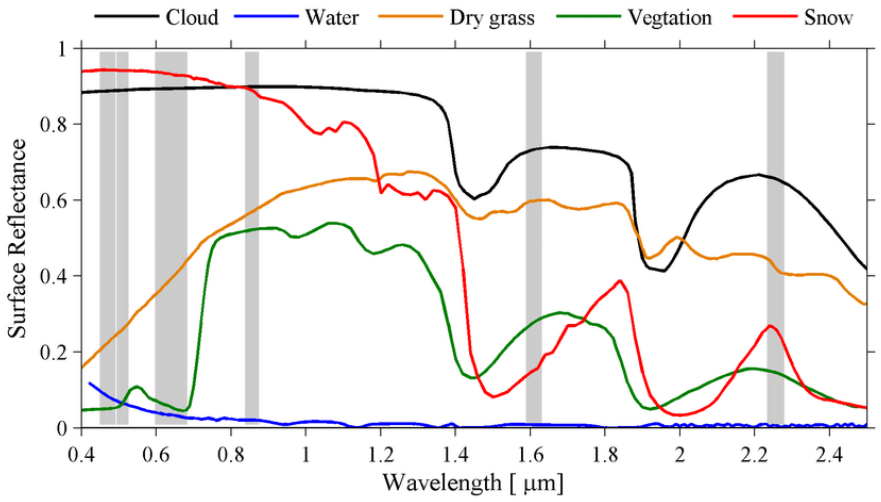


Figure 1.9 Spectral signatures as functions of wavelength for five typical surfaces

The basic concept of the spectral response (that is unique for a given object/surface) is that different objects reflect a different amount of energy in different wavelengths of the electromagnetic spectrum (Figure 1.7). This depends on the properties of material (structural, chemical, and physical), surface roughness, angle of incidence, intensity, and wavelength of radiant energy.

In remote sensing, sensors on board platforms can also be distinguished for varying sensitivities to EM at different wavelengths along the electromagnetic spectrum, in multispectral remote sensing these are generally called spectral bands (or spectral resolution). The spectral resolution is the ability of a sensor to measure the radiation reflected and/or emitted by the surface in a given range of wavelengths.
Sensors can be divided into:

- Multi-spectral: several broad bands centred over the major features of the spectral response;
- Hyper-spectral: hundreds of very narrow spectral bands throughout the visible, near-infrared, and mid-infrared portions of the electromagnetic spectrum.

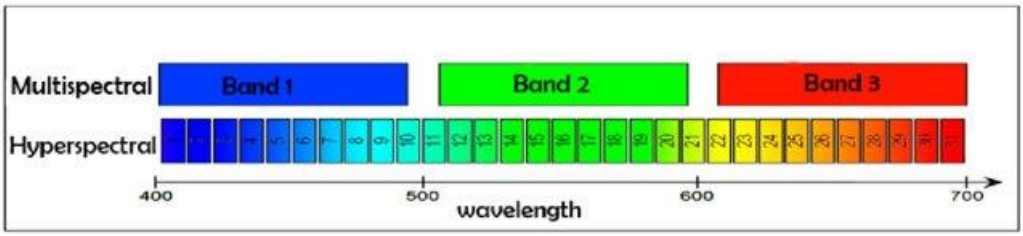


Figure 1.10 Multi-spectral bands vs Hyper-spectral bands in the visible region of EMS

This thesis refers mainly to the imaging systems on board satellite platforms (also called Earth Observation missions) carrying passive multi-spectral sensors.
In particular, it is focused on the use of images acquired by Copernicus Sentinel mission.

1.1.1. Sentinel-2 mission

Copernicus is the Earth observation component of the European Union's Space programme, looking at our planet and its environment to benefit all European citizens. It offers information services that draw from satellite Earth Observation and in-situ (non-space) data [11]. This initiative is headed by the European Commission (EC) in partnership with the European Space Agency (ESA).

ESA has developed a new family of EO missions called Sentinels (Figure 1.11) specifically for the operational needs of the Copernicus programme. Each Sentinel mission is based on a constellation of satellites to provide robust datasets for Copernicus services [12].

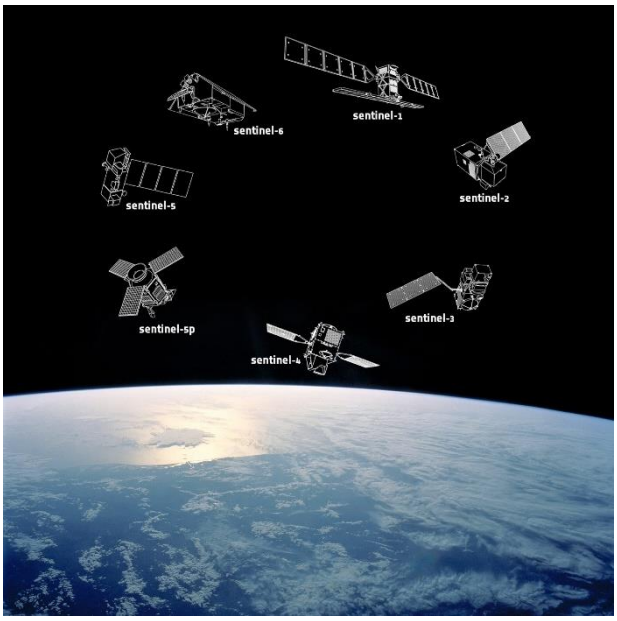


Figure 1.11 ESA Sentinels mission

Specifically, the Sentinel-2 mission consists of two identical satellites: Sentinel-2A and Sentinel-2B. Multi-spectral sensors onboard these two missions have been designed for land monitoring to provide, for example, imagery of vegetation, soil and water cover, inland waterways and coastal areas. Sentinel-2A was launched on 23 June 2015 and Sentinel-2B on 7 March 2017.

Figure 1.12 ESA Sentinel-2 satellite

The two Sentinel-2 satellites operate simultaneously, phased at 180° to each other, in a sun-synchronous orbit at a mean altitude of 786 km.
The two satellites are equipped with MultiSpectral Instrument (MSI). The MSI works passively, by collecting sunlight reflected from the Earth surface. New data is acquired (Table 1.01) by the instrument as the satellite moves along its orbital path. The MSI sensor is a push-broom sensor with 13 spectral bands: four bands at 10 m, six bands at 20 m and three bands at 60 m spatial resolution. In particular, the MSI bands are distributed over the Visible, Near-Infra-Red (VNIR) and Short Wave Infrared (SWIR) wavelengths. The spectral separation of each band into individual wavelengths is accomplished by stripe filters mounted on top of the detectors [13].

Band Number	Band Description	Wavelength Range [nm]	Resolution [m]
B1	Coastal Aerosol	433-453	60
B2	Blue	458-523	10
B3	Green	543-578	10
B4	Red	650-680	10
B5	Red-edge 1	698-713	20
B6	Red-edge 2	733-748	20
B7	Red-edge	773-793	20
B8	Near Infrared	785-900	10
B8A	Near Infrared narrow	855-875	20
B9	Water vapour	935-955	60
B10	Shortwave infrared/Cirrus	1360-1390	60
B11	Shortwave infrared 1	1565-1655	20
B12	Shortwave infrared 2	2100-2280	20

Table 1.01 Sentinel-2 bands acquired and bands details

The Sentinel-2 products can be freely acquired accessing the Sentinel Hub [14]; satellite imagery can be downloaded as multi-band raster images. A band is represented by a single matrix of cell values, also called pixel (picture elements), and a raster with multiple bands contains multiple spatially coincident matrices of cell values representing the same geographic area. Each band represents the intensity of a segment of the electromagnetic spectrum collected by a sensor.

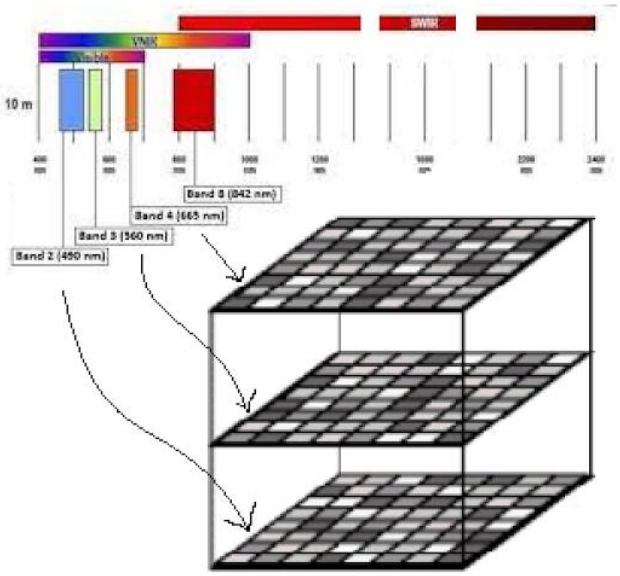


Figure 1.13 Collected EM values for S-2 bands (resolution 10m) stored in multi-bands raster

Different S-2 products are available for users depending on the processing level [15].

- Level-1B: Top-Of-Atmosphere radiances in sensor geometry;
- Level-1C: Top-Of-Atmosphere Reflectances in cartographic geometry;
- Level-2A: Atmospherically corrected Surface Reflectances (SR) in cartographic geometry.

S-2 data are distributed for elementary granules of fixed size (tile), along with a single orbit, with all spectral bands. The tile is the minimum indivisible partition of a product; S-2 images provided in the UTM (Universal Transverse Mercator) projection system that divides the Earth's surface into 60 zones. Tiles can be fully or partially covered by image data; those tiles that are only partially covered belong to the edge or top/bottom of the orbits.

In this thesis work, Level-2A S-2 products were used as input to the burned area mapping algorithm. This product is accompanied by the Scene Classification Layer (SCL) that provides information at the pixel level on the presence of clouds, snow and cloud shadows and generation of a classification map, which consists of three different classes for clouds (including cirrus), together with six different classifications for shadows, cloud shadows, vegetation, not vegetated, water and snow.

Label	Classification
0	No data
1	Saturated or defective
2	Dark area pixels
3	Cloud shadows
4	Vegetation
5	Not vegetated
6	Water
7	Unclassified
8	Cloud medium probability
9	Cloud high probability
10	Thin cirrus
11	Snow

Table 1.1 The classes assigned to each pixel of the S-2 SCL layer in the Level 2A product

1.2. GIS

According to ESRI "a geographic information system (GIS) is a system that creates, manages, analyzes, and maps all types of data. GIS connects data to a map, integrating location data ("where things are") with all types of descriptive information ("what things are like there"). GIS helps users understand patterns, relationships, and geographic context. The benefits include improved communication and efficiency as well as better management and decision making [16].

The GIS environment is composed by the following components, which interact with each other (Figure 1.14):

- methods: a successful GIS operation required the best business rules and excellent setup. These models and operating practices are unique to each organization. These rules can include things such as analysis and how specific analysis is performed. Methods including, what steps followed and in which order;

- people: GIS technology is of limited value without the people who manage the system and to develop plans for applying it. GIS users range from technical specialists who design and maintain the system, to those who use it to help them do their everyday work;
- data: data considered are geographic data and related tabular data that can be collected in-house or bought from a commercial data provider. Most GIS employ a DBMS to create and maintain a database to help organize and manage data.
  The data that a GIS operates on consists of any data bearing a definable relationship to space, including any data about things and events that occur in nature;
- software: set of instructions, data or programs used to operate computers and execute specific tasks like: store, analyze and display of geographical information data;
- hardware: all the equipment that is needed to perform GIS related tasks (pc, server, plotter, GPS, sensor…).

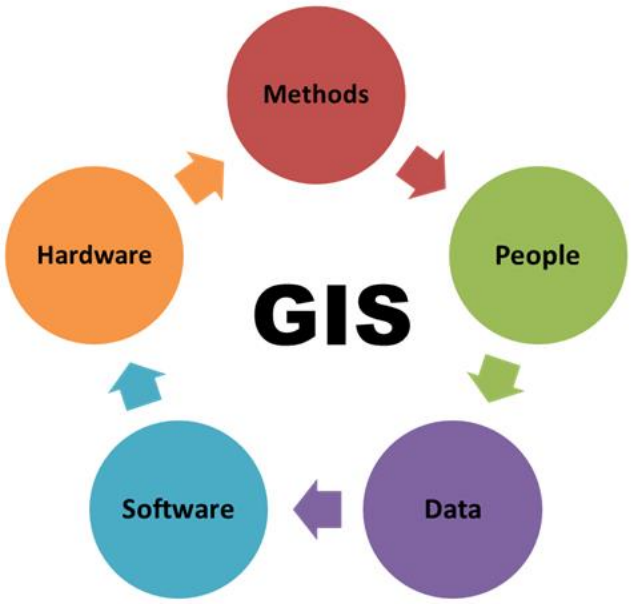


Figure 1.14 GIS components

During this thesis work is used a free and open-source cross-platform desktop geographic information system software that supports viewing, editing, printing, and analysis of geospatial data: QGIS [17].
It can be downloaded and installed following the guide inside the official site.

The software manages and supports different file format, the main are:

- raster (.tif, .jpeg);
- vector (.shp);
- mesh layers (.nc).

Data considered during the are multi-band raster images that come from the Copernicus Sentinel-2 mission.
Furthermore, QGIS gives accessibility to Web Services like Web Map Service and Web Feature Service.
QGIS is a constantly evolving software: new features and new tools are continuously produced and made available to users. One of the reasons for this aspect is the huge community in the GIS environment; in fact every user can contribute by producing new plugins or improving the already existing ones. Obviously a plugin, in QGIS, before being available to the community must be tested and approved by a staff member.

The development of QGIS started in 2002 by Gary Sherman, at the time the project was called Quantum GIS. Only seven years later, in 2009, the version 1.0 of QGIS was published [18].
QGIS is mainly written with these scripting languages:

- C++;
- Python;
- Qt.

The creation/development of a new plugin is possible exploiting these languages. In particular Python and Qt will be used during the development of this work.
This will ensure that the result of this thesis work (Burned Area Detector - BAD) will be accessible and free to every user.

1.3. Python

Python was born in the late 1980s by the idea of Guido van Rossum to offer to the user a high-level, general-purpose, free, open-source programming language, and therefore accessible to everyone and quite intuitive. In fact, its design philosophy emphasizes code readability with the use of significant indentation [19].

This philosophy can be read in the document "The Zen of Python (PEP 20)" [20], which includes aphorisms such as:

- beautiful is better than ugly;
- explicit is better than implicit;
- simple is better than complex;
- complex is better than complicated;
- readability counts.

In order to develop the BAD plugin the following Statement and control flow are used:

- if, else: code inside this statement is executed if the condition is true, otherwise (else) if the condition if false is executed the code under else block;
- for: code inside this statement is repeated many times is specified;
- while: code inside this statement is repeated until a specific condition is met;
- class: block of code is executed and attaches its local namespace to a class, for use in object-oriented programming;
- def: statement that defines a function or method;
- import and from: statement used to import modules whose functions or variables can be used in the current program.

Later on, in Chapter 4, will be better described how these elements are used and "for what".
During the development of the code the following data structure are used:

- number variable (integer, float);
- string variable (characters in quotation marks);
- list: it is used to store multiple items (any type of variable) in a single variable. Elements can be added (appended) and/or removed to the list.
- array/ndarray: Python does not have built-in support for arrays. In order to work with this structure it is necessary to import a specific library: Numpy.
- Thanks to the library it is possible to use this structure in order to manage vectors and matrices and tensors:

- vector: 1-Dimension array;

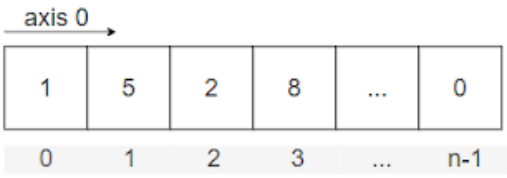


Figure 1.15 Array with shape (n), axis and indexes

- matrix: 2-Dimension array;

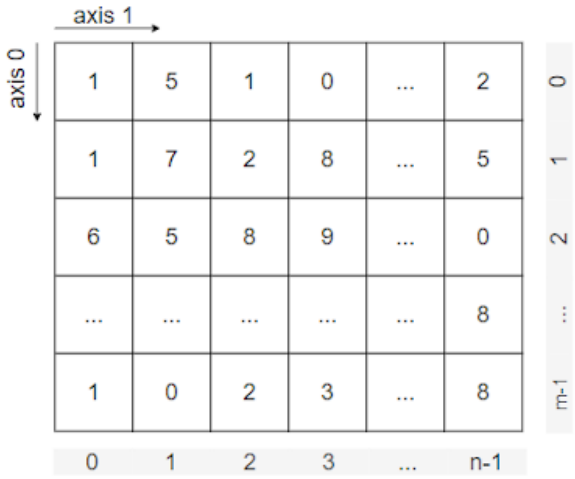


Figure 1.16 Array with shape (n,m), axis and indexes

- tensor: N-Dimension array;

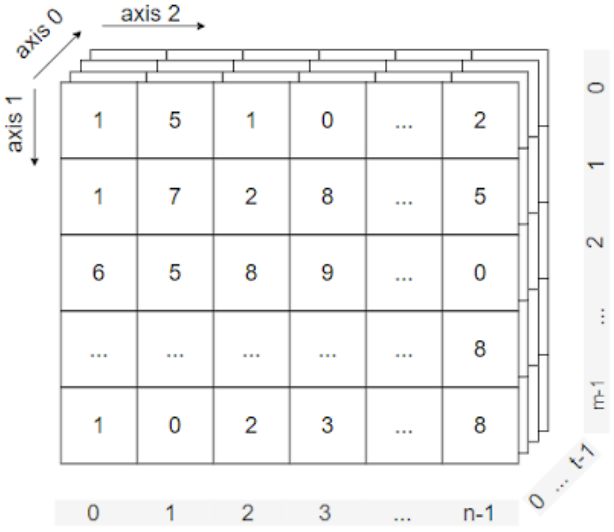


Figure 1.17 Array with shape (n,m,t), axis and indexes

Array/Ndarray is the fundamental structure for developing the BAD plugin, because it allows to compute operations on input bands and manages the input/output products like matrix/tensor.

For those who also use different programming languages, one of the major sources of error reading and writing code in Python, is the following one: Python starts to count from 0, so, the first element of an array, a list, a matrix etc has index equal to 0 and not to 1.

2 REMOTE SENSING OF FIRES

Remote Sensing techniques can be used to monitor the phenomena of wildfire from global to local scale, because fires produce anomalies that are detectable in different wavelengths of the electromagnetic spectrum. The spectral signature of burned areas is determined by changes induced by the fire on the vegetation; these changes are mainly observable in the near infrared and shortwave infrared wavelengths.
When the phenomena of wildfire is considered different products can be derived from EO multi-spectral data:

- active fires;
- burned areas;
- burn severity;
- smoke plumes.

The first three are the most commonly used for monitoring the fire occurrence at various spatial and temporal scales of observation. Identifying and mapping smoke plumes, on the contrary, is used to detect wildfire phenomena at local scale as an alternative to visual detection by human operators, therefore it will not be considered during this work [21].

Figure 2.1 Plumes of smoke rise above Frenchman Lake as the Sugar fire, part of the Beckwourth Complex, burns in Plumas National Forest on July 2021 [22].

2.1. Active Fire

A first indicator of the presence of a fire is the detection of active fire (flaming front of the fire).

Figure 2.2 front of the fire in Karbole outside Ljusdal, Sweden, on 2018-07-15 [23]

According to Copernicus [24] active fires are identified on the basis of the so-called thermal anomaly produced by them. The algorithms compare the temperature of a potential fire with the temperature of the land cover around it; if the difference in temperature is above a given threshold, the potential fire is confirmed as an active fire or "hot spot".
The data related to active fire are distributed by the NASA FIRMS (Fire Information for Resource Management System) that uses satellite observations from two possible sensor:

- MODIS (Moderate Resolution Imaging Spectroradiometer) sensor: is mounted on TERRA and AQUA satellites; it is the base instrument in order to detect active fire in near real-time (NRT). The anomaly (or anomalies) is detected by the Fire and Thermal Anomalies algorithm (Giglio 2003) [25] and each 1 km x 1 km pixel identified as thermal anomaly contains information on fire occurrence (day/night), fire location, confidence level, Fire Radiative Power and other layers describing fire pixel attributes. All information are available in the official site [26].

- VIIRS (Visible Infrared Imaging Radiometer Suite) sensor: it is on board of the NASA/NOAA Suomi National Polar-orbiting Partnership (SNPP). It works like the MODIS sensor, but with the advantages of a greater spatial

resolution of 375 m. The enhanced spatial resolution allows the detection of smaller fires and to outline the shape of ongoing large fires.
Much more information is available on the official website [27].

The data related to active fire can be consulted, and also downloaded, from the NASA FIRMS (Fire Information for Resource Management System) site [28].

2.2. Burned Area

The area burned is the area affected by a fire that shows a significant change in the vegetation cover; the vegetation layer is indeed damaged by the fire and the effect can be observed by RS as destruction of dry material and reduction or loss of green material; immediately close to the fire event, the RS signal could be also influenced by the presence of ash (darker surface).

Figure 2.3 Area damaged by the Bootleg Fire smolders near the Northwest edge of the blaze on 2021-07-23, near Paisley, Oregon [29]

Fires produce a significant change in the structure and the reflectance of vegetation and the soil properties within the burnt area that are noticeable in the microwave, visible and especially the infra-red wavelengths of the electromagnetic spectrum (Figure 2.4 (a) and (b)).

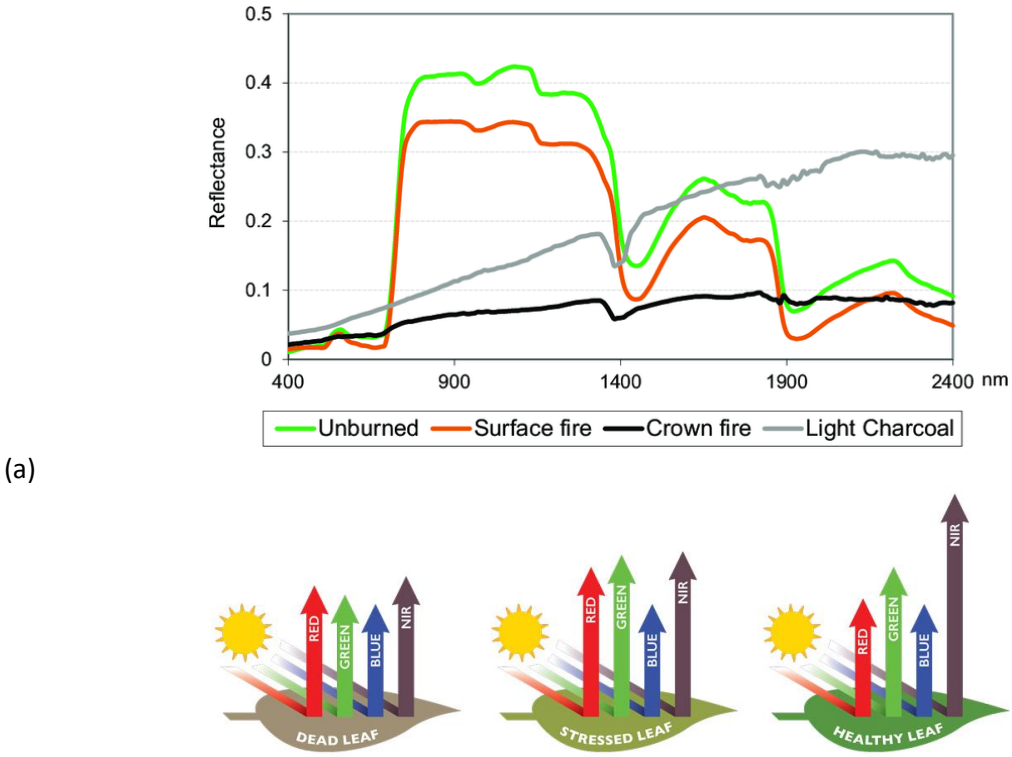


Figure 2.4 Reflectance spectra for unburned vegetation canopy and fires affecting different vegetation strata. Spectra were simulated using Prospect+Geosail models [30] *(a); Vegetation Reflectance (b)*

In fact, healthy vegetation has high reflectance in the NIR, and low reflectance in the SWIR portions of the spectrum (Figure 2.4 (a)) - the opposite of what happens in areas affected by fire, where vegetation has been burned and damaged. The use of satellite images, for mapping burned areas at regional to global scales, has a long history in RS studies starting in the early 1970s and 1980s [30] and it is still an active research topic employing advanced techniques that integrate geo-statistics, object oriented and machine learning methods.

Both active fires and burned area mapping is undertaken with remotely sensed data although burned area detection is generally a more suitable indicator for the assessment of the damage induced by wildfire since it is more persistent in time. The temporal persistence of these changes is highly variable, ranging from a few weeks in savannas to years in forests (Melchiorre and Boschetti, 2018 [31]). The basic idea is to produce a Burned Area (BA) Map to represent the surfaces which have been affected by fires and to identify whether a pixel is burned or not.
The identification of burned conditions at the pixel scale can be combined with

severity mapping to quantify the level of damage occurred to the vegetated layer (see 2.3).
Since the earliest satellite missions for Earth Observation applications, a wide range of techniques and algorithms have been developed and applied for BA mapping, obviously the evolution of these methods strictly depends on the new technology developed and used.
According to the review article by Koutsias et al. (1999) [32] classification approaches to Burned Area mapping can be classified in three general groups depending on:

- multi-temporal or single date satellite images are employed;
- the output is a direct estimate of BA or an intermediate enhanced product;
- the type of classification methods.

Multi-temporal approaches have the advantage over single post-fire images of reducing commission errors (to report an information which is not true) caused by dark soils, water bodies, topographic shades, or cloud shadows that might be spectrally similar to burned areas. Therefore, the BA detection utilizes information not only from spectral, but also from temporal changes between the pre-fire and post-fire satellite imagery.
However, there are some difficulties related to radiometric and geometric adjustments, as well as to the discrimination of fire-caused changes from other types of temporal changes, such as seasonal floods, harvesting or deforestation.

The second group of the techniques reduces the dimensionality of the original images. This is the case of principal component analysis (PCA has been used since early 1980s for BA and change detection analysis (Richards, 1984 [33])) or vegetation indices, which aim to improve spectral separability of burned versus other covers.
Thresholding of spectral vegetation indices are also very common techniques for BA discrimination.
Among the several spectral indices that are used for burned area mapping, the most common are:

- the Normalized Burn Ratio (NBR);
- the Normalized Difference Vegetation Index (NDVI);
- the Soil Adjusted Vegetation Index (SAVI);
- the Global Environment Monitoring Index (GEMI) and the Burned Area Index (BAI).

In multitemporal approaches, the spectral signals of pre-fire and post- fire images, in some specific bands(channels), are usually compared, to identify the sharp changes due to fires.

Finally, the third group of studies refers to the classification techniques:

- supervised;
- unsupervised.

The choice of the method depends on the availability of previous knowledge of fire effects in the target region. The maximum likelihood classification and k-means clustering algorithms were employed in several studies either to directly map BA, or to evaluate other classification techniques (Henry 2008; Pereira and Setzer, 1993 [34]). Methods based on logistic regression were introduced to map BA using multi-date (Koutsias and Karteris, 1998 [35]) and single-date (Koutsias and Karteris, 2000 [36]) Landsat TM imagery. The main consideration when implementing BA classifications was to express the classification problem in a binary way: burned vs unburned pixels.

Both active fire and burned area mapping algorithms have also exploited another technique based on is the Region Growing (RG). RG is a region-based contextual image segmentation method, that iteratively adds pixels from neighboring regions of an initial segmentation and/or a set of pixels based on a set of conditions. Starting from pixels called seeds the approach examines the neighboring pixels and, if specific constraints are true, the pixel neighbors are added to the region becoming new seeds for the next iteration. The process is iterated on, in the same manner as general data clustering algorithms.
This approach is considered during this thesis work, therefore more detail about the algorithm, how seeds are selected, which are the constraints that must be considered and how the algorithm is implemented is described in the following chapters (Chapter 3 and 4).

Machine learning algorithms have been largely exploited for burned area mapping thanks to their ability in managing large datasets; common machine learning algorithms are: random forests, support vector machine, artificial neural networks and decision trees (Ramo et al., 2018 [37]).

For this thesis work, it is considered the algorithm developed by CNR-IREA, that is well described in the article "*A Burned Area Mapping Algorithm for Sentinel-2 Data Based on Approximate Reasoning and Region Growing*" (Sali et al. 2021 [2]).
The CNR-IREA algorithm proposed is a classification algorithm for S-2 imagery exploiting pre-fire and post-fire acquisitions (multi-temporal approach) to maximize mapping accuracy.
The algorithm exploits the multi-criteria soft aggregation approach of burn evidence already proposed by Stroppiana et al. (2012) [38].

In the new approach the soft constraints are defined by membership functions of fuzzy sets based on statistics (percentiles) of reflectance as derived from training areas.
Major details and the algorithm specifications are reported in Chapter 3.

Once an algorithm has been developed and tested, the next step is its implementation, in order to produce an instrument, a tool, a plugin that can be delivered to the users community.
The implementation can be done with many different programming languages and/or environments; in this thesis is used Python and QGIS, which are both free and open-source.
The basic rules, when an algorithm is implemented, are make it:

- able to produce valid output;
- automated as much as possible;
- user friendly as much as possible.

If these requirements are satisfied, the tool should be able to process large datasets and/or to have them routinely applied, to give accessibility to users with different background knowledge. This is the ultimate goal of this thesis: to implement a plugin that can provide each user with valid and meaningful output, on a desired area, in the clearest and simplest way possible.

Furthermore, there are plugins, for example in QGIS, that implement general Land Cover (LC) classification algorithms (such as dzetsaka [39]) that can be used to detect BA, which is a particular case of LC classification and change detection.
Instead, by looking at plugins specifically developed for BA mapping, in addition to the one developed by CNR-IREA, in literature the principal are:

- National Observatory of Forest Fires (NOFFI);
- North Australia & Rangelands Fire Information (NAFI) Map Services.

These plugins can be downloaded in QGIS, or can be accessed as Web Service.
The main features and a brief description of the algorithm are reported for completeness in paragraph 3.1.

2.3. Burn Severity

Another product that can be considered when the wildfire phenomena is studied is the Burn Severity. Burn Severity is the assessment of the level of damage to the vegetation induced by a fire.
By definition the Burn Severity describes how the fire intensity (energy that is released from organic matter during the combustion process) affects the functioning of the ecosystem in the area that has been burnt. The observed effects often vary within the area and between different ecosystems (Keeley, 2009 [40]). Burn Severity can also be described as the degree to which an area has been altered or disrupted by the fire.

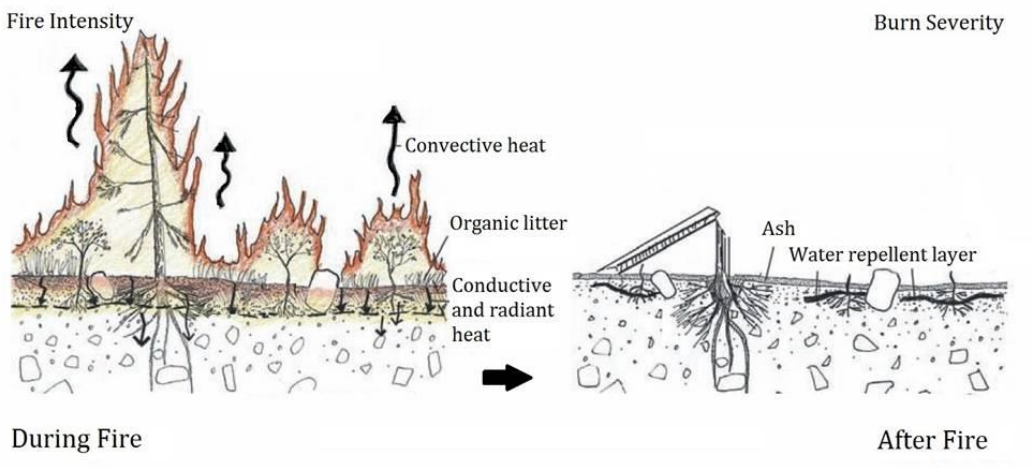


Figure 2.5 Fire intensity vs burn severity

As already described in previous paragraph (2.2), the reflectance of healthy vegetation is different from the burned one. This behaviour can be used in order to compute a vegetation index largely used for monitoring burn severity: Normalized Burn Ratio (NBR).

$$NBR = \frac{NIR - SWIR}{NIR + SWIR}$$

Formula 2.1 Computation of NBR index [41]

As shown in Figure 2.4 (a) the difference between the spectral responses of healthy vegetation and burnt areas reach their peak in the NIR and the SWIR regions of the spectrum, for this reason they are involved in the NBR computation. A high NBR value indicates healthy vegetation while a low value indicates bare

ground and recently burnt areas. Unburned areas are normally attributed to values close to zero.

One of the most used approaches to identify the changes induced by the fire is the computation of the difference of pre-fire and post- NBR values, called Delta NBR (dNBR or ΔNBR) which then can be used to estimate the Burn Severity.

$$dNBR = preNBR - postNBR$$

Formula 2.2 Computation of dNBR

Delta NBR values indicate the level of damage in the affected area:

- a higher value indicates more severe damage;
- value close to zero indicates unburned area;
- lower value (negative) may indicate regrowth following a fire.

In order to properly interpret the result coming from the dNBR values the interpretation should also be carried out through field assessment, because it can vary from case to case.

The United States Geological Survey (USGS) proposed a classification table to interpret the Burn Severity (Table Burn Severity) based on dNBR values. These values are considered in this thesis as default values in order to implement the algorithm for burn severity mapping.

Label	Severity Level	dNBR Range
1	Enhanced Regrowth, High	-0.500 to -0.251
2	Enhanced Regrowth, Low	-0.100 to -0.101
3	Unburned	-0.100 to +0.099
4	Low Severity	+0.100 to +0.269
5	Moderate-Low Severity	+0.270 to +0.439
6	Moderate-High Severity	+0.440 to +0.659
7	High Severity	+0.660 to +1.300

Table 2.1 Burn Severity level [41]

Finally, the Burn Severity Map can be plotted:

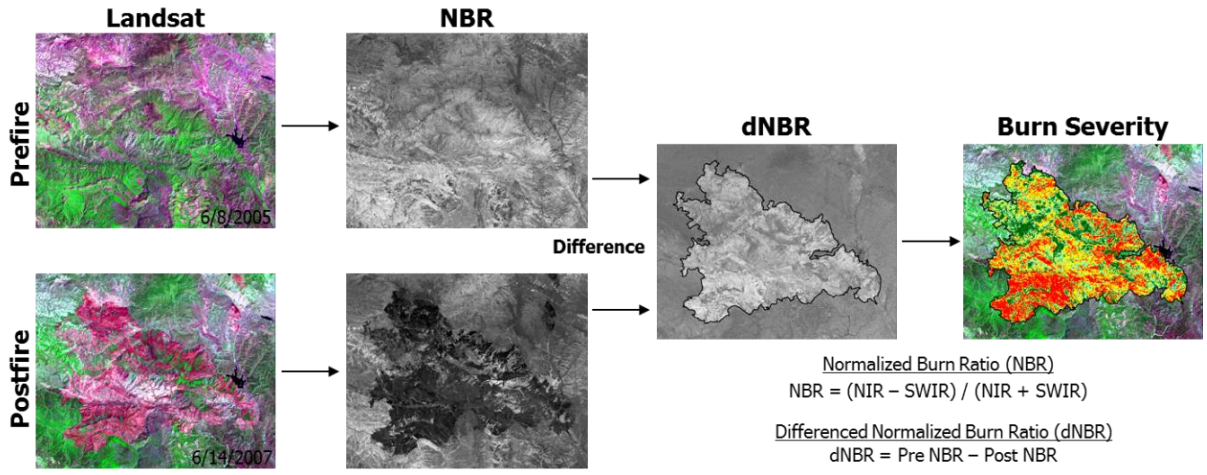


Figure 2.6 Burn Severity mapping process [42]

2.4. Copernicus services for fire monitoring

RS imagery is largely used in the Copernicus Emergency Management Service (Copernicus EMS) that provides timely and accurate geo-spatial information, since 2012, to actors involved in the management of natural disasters, such as fires (but also flooding, earthquakes, etc.) [43].

The Copernicus EMS consists of two components:

- a mapping component. This component of the service (Copernicus EMS-Mapping) has a worldwide coverage and provides maps based on satellite imagery. These maps contain information for selected emergency situations that arise from natural or man-made disasters anywhere in the world.
- an early warning component. This component is mainly devoted to responding to emergencies in real-time and it consists of three different systems:
    - the European Flood Awareness System (EFAS), which provides overviews on ongoing and forecasted floods in Europe up to 10 days in advance;
    - the European Forest Fire Information System (EFFIS), which provides Near Real-Time and historical information on forest fires and forest fire regimes in the European, Middle Eastern and North African regions;
    - the European Drought Observatory (EDO), which provides drought-relevant information and early-warnings for Europe.

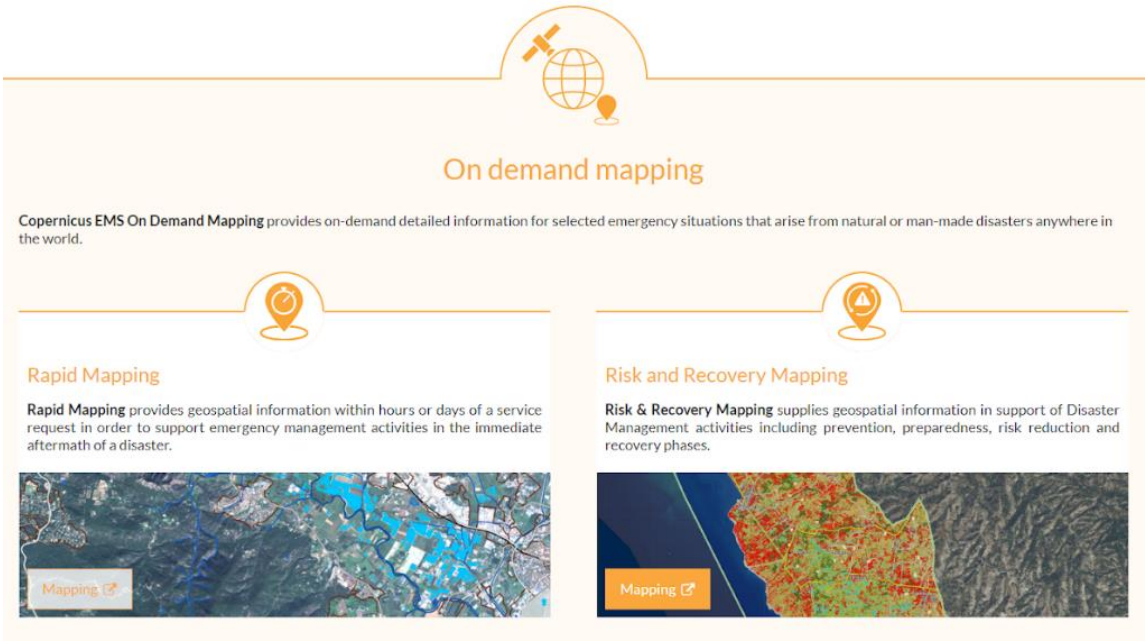


Figure 2.7 Clip of the CEMS Web Site for the mapping component

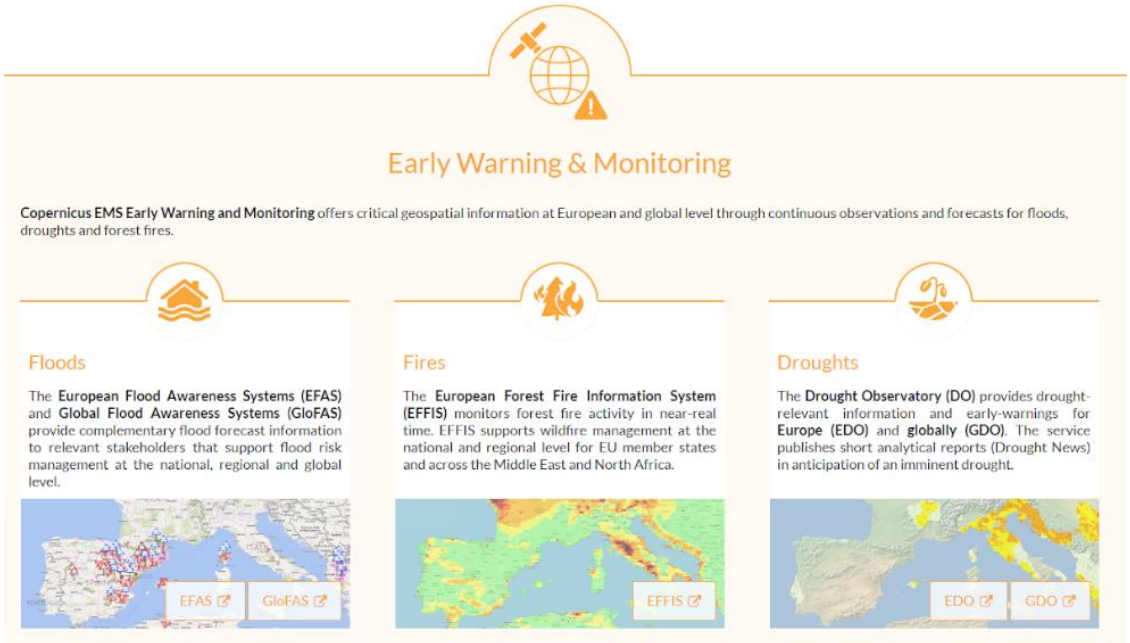


Figure 2.8 Clip of the CEMS Web Site for the early warning component

In this thesis work are taken in account the CEMS maps related to wildfire events. These maps are considered as reference data in order to validate the output of the developed plugin (Chapter 5).

3 CNR-IREA ALGORITHM

The theory behind the BAD plugin is the pixel-based algorithm developed at IREA CNR to map Burned Area (BA) from Sentinel-2 images and based on the theory of fuzzy sets and convergence of evidence (Sali et al. 2021 [2]) The RG module, developed in this thesis, is part of the fuzzy BA algorithm and it is used as a final step to reduce misclassification errors in burned area maps. The flowchart of the fuzzy BA algorithm is given in Figure 3.01 and a brief description of the main steps that lead to the generation of the Burned Area Map (BM) is given for completeness in the following sections.

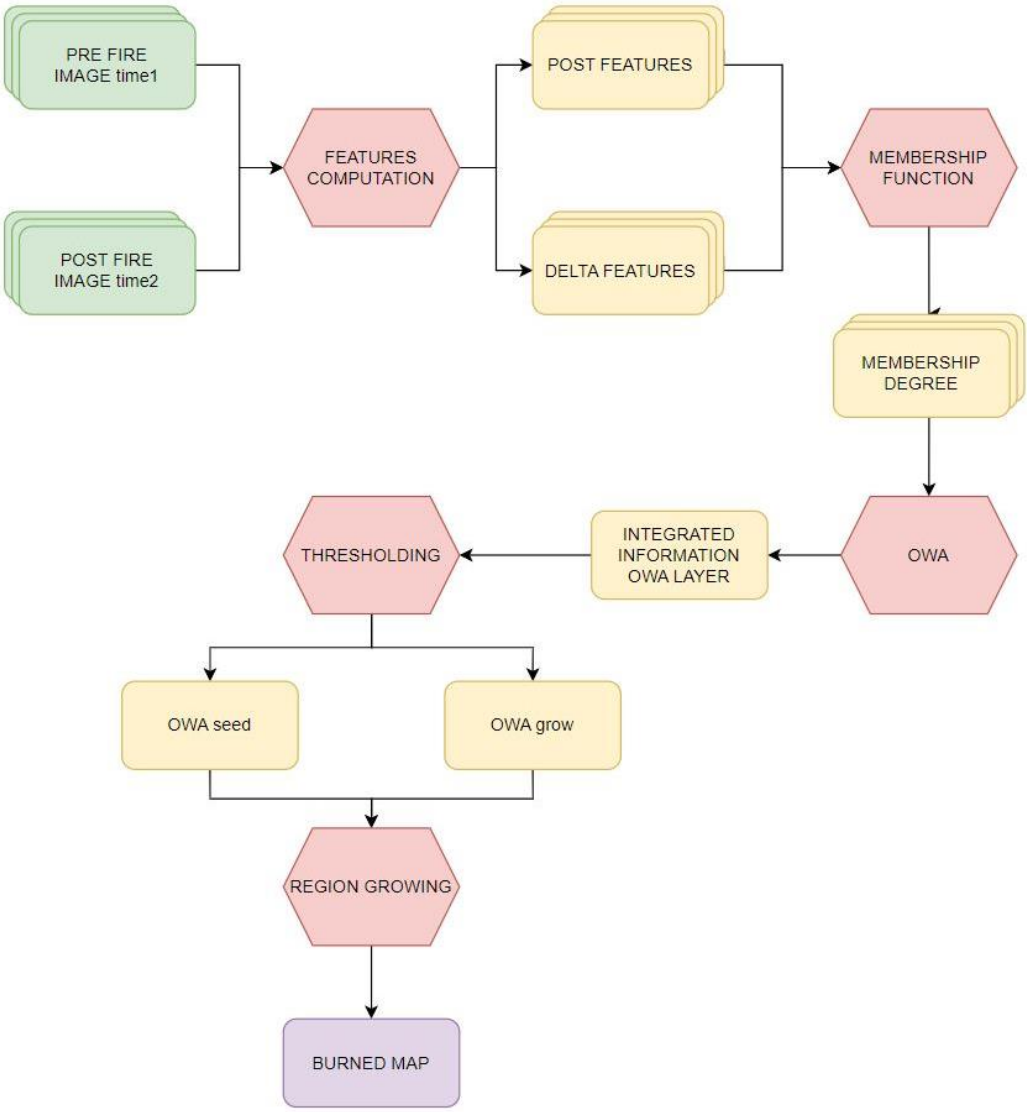


Figure 3.1 Burned Area algorithm flowchart

The principal steps are:

a) Input Feature Selection:

Two multi-spectral satellite Sentinel-2 images of the area of interest, acquired at different time for change detection, are selected. Generally, before proceeding, a mask is applied in order to make a distinction between what may burn (vegetation, label 4) and what cannot burn (lack of vegetation, clouds and cloud and topographic shadows). The mask is applied to run the algorithm only for vegetated pixels that could burn. In this thesis, the mask for identifying burnable surface is derived from the Scene Classification Layer (SCL) available with Level 2A Sentinel 2 image products (Table 1.1):

At this point, for each image, significant bands are selected and differences between them are computed; input S-2 spectral bands and their combination used as input to the classification algorithm are called input features.

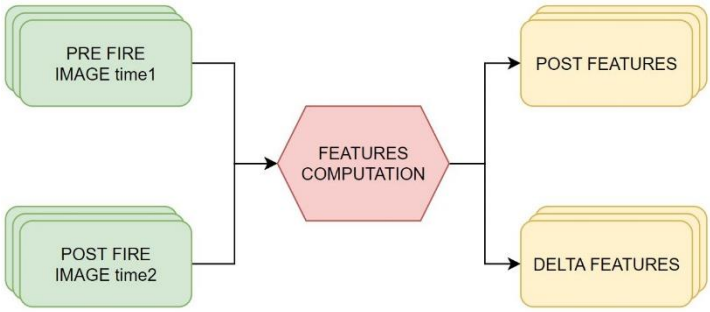


Figure 3.2 Burned Area algorithm flowchart-Feature Computation Input/Output

This step is at the base of the entire process: the selection of most suitable input features (Table 3.1) is done to deliver the best possible classification of the two images, in order to identify the areas affected by fires at different periods. Input images should be selected with the least cloud cover in order to maximize the surface observed.
Images are exploited to identify areas that burned between the two dates (also called pre-fire and post-fire images/dates) by relying on reflectance values and reflectance difference between the two acquisition dates.

Input feature name	Feature computation
post RedEdge	Post image BAND6
post RedEdge	Post image BAND7
post NIR	Post image BAND8
delta RedEdge	Post image BAND6 - Pre image BAND6
delta RedEdge	Post image BAND7 - Pre image BAND7
delta NIR	Post image BAND8 - Pre image BAND8
delta SWIR	Post image BAND12 - Pre image BAND12

Table 3.1 Main input feature and their computation

b) Membership degree computation:

by applying the fuzzy set theory on the results of the previous step, it is possible to define a Membership Functions (MF) for each feature that is applied to each pixel. MF assigns to each input (domain of the function) a score (value inside the codomain of the function). The assigned score value is called Membership Degree (MD). This value belongs to the following interval: [0,1] (0 and 1 included). MD quantifies the grade of membership of the element with respect to the considered category, in this work Burned category. It means that values closer to 1 are more likely Burned (B), conversely MD closer to 0 means that the pixel has low degree to belong to Burned category.
If the degree of membership to the Burned category is low (MD closer to 0), the pixel is likely to be Unburned (U).

In the fuzzy BA algorithm, the function considered is a sigmoid:

$$MF(x) = \frac{L}{1+\exp(-k(x-x0))}$$

Formula 3.1 Membership Function Sigmoid

where L is the upper limit of the function, in this case L→1 to quantify the maximum degree of membership, K is the curve's slope and x_0 is the inflection point.

Depending on the K value the MF can assume different shape:

- if K < 0, MF assumes a z-shape;
- if K = 0, MF is constant to 1/2;
- if K > 0, MF assume a s-shape;

For example, in the NIR S-2 reflectance band, fire occurrence leads to a decrease in the pixel's reflectance compared to unburned vegetation, hence a z-shaped function was used for the post-fire NIR feature.
On the contrary, an s-shaped function is used when fire occurrence leads to an increase of pixel reflectance. The same applies for the delta features: difference between
pre-fire and post-fire reflectance values.

The two parameters K and x_0 are estimated by using percentiles of the frequency distribution of features' values for the burned area class.

Based on Roteta et al. (2019) [44], a different shape is selected for the sigmoid functions depending on the spectral response of burned areas in the domain of the specific input feature: for the selected features the fuzzy BA algorithm applies a s-shaped function or z-shaped function (Figure 3.4).

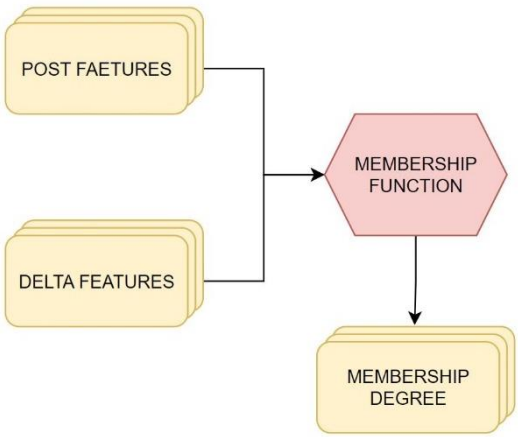


Figure 3.3 Burned Area algorithm flowchart-Membership Function Input/Output

MFs are applied to each pixel to compute the degree of membership (MD), that is a continuous value between 0 and 1 that represents the membership degree of each pixel to the burned category:

- the value obtained tends to 0: it means that the pixel has low degree of membership to be Burned, this is given by the 90th and 10th percentile of the

frequency distribution function of the unburned training pixels for the s-shaped and z-shaped function, respectively (Figure 3.4);

- the value obtained tends to 1: it means that the pixel is more likely to be Burned (B), this is given by the 50th percentile of the frequency distribution function of the unburned training pixels for the s-shaped and z-shaped function (Figure 3.04).

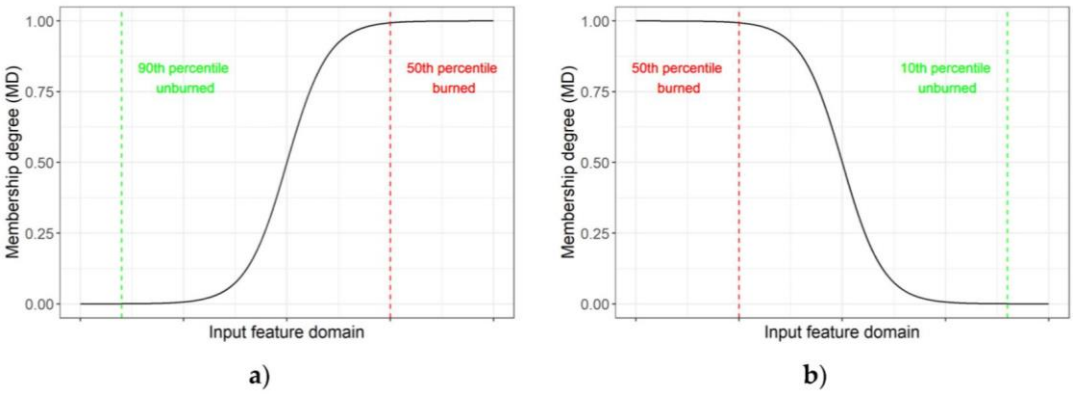


Figure 3.4 Sigmoid function s-shape (a); Sigmoid function z-shape (b)

c) Ordered Weight Averaging (OWA) computation:

OWA operators are exploited to integrate MDs from different features into a synthetic score; these operators implement the convergence of evidence (agreement from redundant information/observations provided by the N input features).
In this thesis work, N=7 (see Table 3.1).

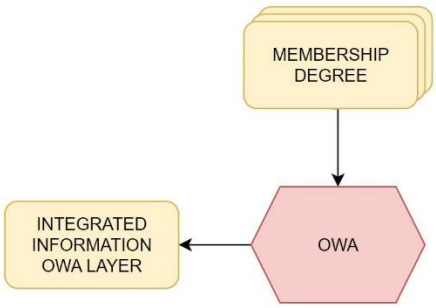


Figure 3.5 Burned Area algorithm flowchart-OWA Input/Output

The synthetic score can then be converted to binary information Burned/Unburned, if the final result should be a binary map in which each pixel contains only one information: Burned or Unburned. The synthetic values could also be retained as continuous and correlated, for example, to the level of damage (i.e., burn severity).

Indeed, up to step b), it is computed a continuous value of MD for each pixel and input feature (n); so, it is necessary to convert a multi-dimensional information to a single layer (a sort of synthetic information on the likelihood of begin burned for each pixel), that is the global evidence of burn. In order to do this, OWA operators are used to integrate input information derived from multiple features for each pixel, by applying a weighted average operator. The weighted average is applied after sorting input MD values for each pixel in descending order. In the fuzzy set theory, OWA operators belong to a parameterized family of soft-mean-like aggregation operators [45]. In the fuzzy BA algorithm, different operators were used to represent attitudes ranging between:

- pessimistic/maximum ($OWAor$ considers the maximum extent of the phenomenon to minimize the chance of underestimating the area burned: modelling a compensative aggregation to integrate complementarities of multiple criteria);
- optimistic/minimum ($OWAand$ considers the minimum extent of the phenomenon to minimize the chance of overestimating the area burned: modelling a concurrent aggregation to integrate mutual reinforcement of multiple criteria).

Other OWA operators are: OWA almostOR, OWA average, OWA almostAND.
How to compute each of them is described in Table 3.2.

OWA operator	Weight array	Function
AND	[0,0,0,.....,0,0,1]	min
almost AND	[0,0,0,.....,0,0.5,0.5]	(min+penultimate)/2
AVERAGE	[1/n,1/n,1/n,...,1/n,1/n,1/n]	mean
almostOR	[0.5,0.5,0,.....,0,0,0]	(max+second)/2
OR	[1,0,0,.....,0,0,0]	max

Table 3.2 OWA operator table

Layers of global evidence derived with different $OWAs$ will be input to the region growing algorithm.

d) Region Growing:

this last step is based on two different layers: Seed and Grow. The Seed layer in a RG algorithm represents the first initial partition of the image that is composed of those pixels that are most likely to be burned. In the fuzzy BA algorithm, the seed layer is generated by computing the $OWAand$, which is the most conservative operator (implementing a minimum criterion) and it tends to select only pixels with the greater likelihood of being burned (greatest MD value) and to reduce commission errors.
The Grow layer is used to iteratively expand seed pixels; the grow layer is obtained from the $OWAor$, and it allows to reduce omission classification errors by adding pixels that are less likely burned only if they are neighbors of the more likely burned pixels (seeds).
When these two layers are obtained, a different threshold is applied to both of them. This is done to set the seeds inside the Seed Layer, and to individualize which pixel can grow inside the Grow Layer.

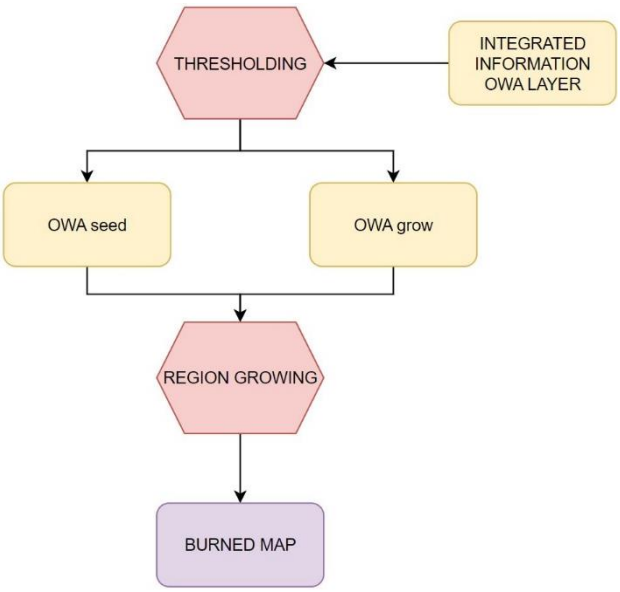


Figure 3.6 Burned Area algorithm flowchart-Region Growing Input/Output

These two layers combined in a contextual region growing algorithm can provide the most accurate results in terms of burned area mapping. As the name suggests the basic idea is to make seeds grow on the growing layer.
The goal of this thesis work is to develop a plugin (BAD) that should be able to analyze the input multi-band raster images and provide information about the wildfire phenomena (Burned Area Map, Severity Map) using the algorithm just described.

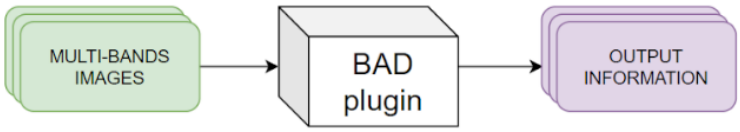


Figure 3.7 Input/Output BAD plugin

3.1. Others tools

The North Australian Fire Information (NAFI) Website, as reported in the official document [46] and in FAQ document [47], provides maps of fire based on satellite images to help fire managers across regional and remote northern Australia. In addition to the Website it is possible to download and install the plugin in QGIS.
The service provides critical Near Real-Time information on active fire, and burnt area mapping that are continually updated when new information are available.
These data are produced via a number of satellites that pass over Northern Australia multiple times every day:

- active fire information are sourced from Landgate Western Australia (from NOAA and NASA satellites) and Geoscience Australia (from NASA satellites);
- burnt area information are derived from MODIS (Moderate-resolution Imaging Spectroradiometer) satellites.

Active fires are shown via hotspots which are updated every few hours. Hotspots are produced from thermal (heat) sensors on a number of different satellites. They are usually accurate to within 1km of their actual location.

Burnt area maps show to the user what has already burnt, and are updated every week or so throughout the entire year. They are produced by comparing two different satellite images (multi-temporal approach), generally 1-2 weeks apart, and identifying only the areas that have been burnt.

The maps are produced using a change-detection based on two input images that are acquired on a different date.

The National Observatory of Forest Fires (NOFFi), as reported inside the official page [48], aims to develop a series of modern products and services for supporting the efficient forest fire prevention management in Greece and the Balkan region, as well as to stimulate the development of transnational fire prevention and impacts mitigation policies.

NOFFI offer three main different products about wildfire phenomena:

- a RS-based fuel type map;
- a semi-automatic burned area map;

- a dynamically updatable fire danger index providing mid-term predictions.

The first product (fuel type map), as reporter in the article "The Greek National Observatory of Forest Fires (NOFFi)" [49] allows the user to visualize the type of fuel present in the area analyzed.
The fuel type categories that are considered are:

- level 1: agricultural areas, urban areas;
- level 2: non-vegetated areas, waterbodies;
- level 3: deciduous broadleaves (three density classes: 0–40%, 41–70%, and 71–100%), coniferous (three density classes as before), evergreen broadleaves (three density classes as before), grasslands and shrublands.

In order to assign a category to an area (pixel) fuzzy logic rules are used. These rules are constructed by selecting a subset of features.
The data that are considered are collected from:

- winter and summer Landsat 8 OLI images acquired during 2014-15;
- object-based classification approach;
- accuracy assessment based on the LUCAS 2013 survey.

The second service offered by NOFFI is the burned-area map computation. The map is obtained applying an object-based image analysis approach and state-of-the-art supervised classification models. The service is delivered as a set of custom executable programs written in the C++ programming language, which perform the necessary individual processes. Nevertheless, user interaction is performed through a QGIS plugin. Data used in order to compute the map come from Landsat 8 OLI and S-2.

The final product is the fire danger index:

- 1=low;
- 2=moderate;
- 3=high;
- 4=very high.

In order to compute the index, the following attributes are considered:

- processing of MODIS satellite time-series for the estimation of vegetation dryness;
- spatial estimation of dry fuel connectivity;
- combination of the derived satellite-based measurements with fuel type, topography, and land use through a multi-criteria analysis scheme for the estimation of fire danger.

4 PLUG-IN IMPLEMENTATION

In this chapter it is described step by step how the plugin, Burned Area Detector (BAD), was implemented and how it works.
The idea that has led the entire development of the plugin is the following: to give to the user an instrument easy to use, intuitive to be managed and at the same time flexible to leave the user free to choose the input features, the parameters of the MFs. BAD can be executed as a whole (from input S-2 image to burned area maps) or step by step based on the user's needs.
Indeed, the plug-in has main modules that can be executed either completely independently of each other or in a batch processing.

Figure 4.1 TABs of BAD plugin

In Figure 4.1 is reported the most general structure of the plugin. Each element represents a TAB in BAD.

Before going into the details of the plugin, it is necessary to introduce three tools that was used in order to develop BAD:

- Plugin Builder 3;
- Qt Designer;
- PyQt5.

4.1. Plugin builder 3

Plugin Builder3 is a plugin inside QGIS. It can be installed directly from the QGIS interface by clicking on the top bar on the Plugins section and searching for: "*Plugin Builder 3*". In this thesis, version 3.2.1, developed by GeoApt LLC, was used [50]. When it is installed, it appears on the top bar of QGIS  . At this point it is ready to be used.

The QGIS Plugin Builder allows to create different types of templates that are the starting point in plugin development.

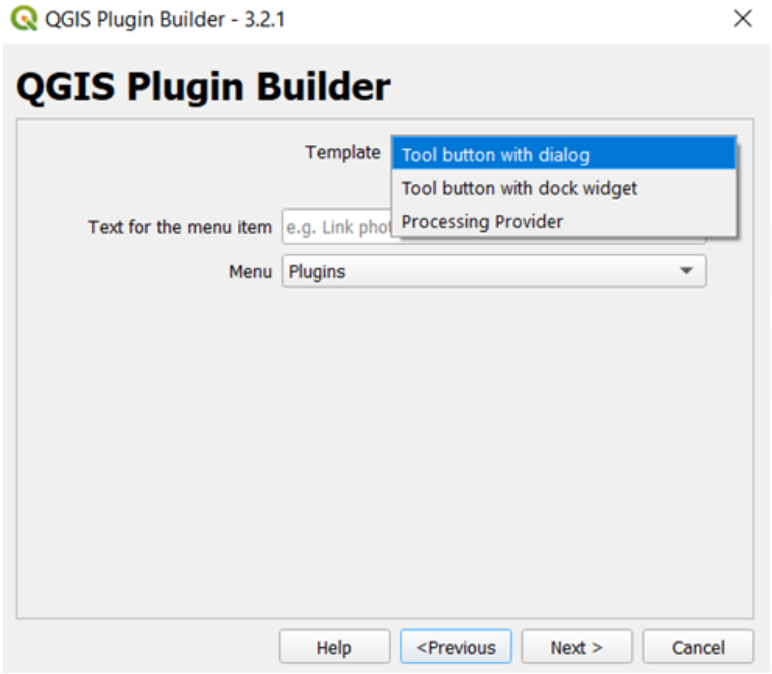


Figure 4.2 Plugin Builder 3 User Interface

The plugin was realized by selecting the first template type: "*Tool button with dialog*": this means that BAD's icon will be present in the plugin menu and when it is clicked a dialog window appears on the screen.

As already mentioned, Plugin Builder 3 gives the starting point to develop a plugin. In fact, after the first step, a folder is automatically generated. In this folder are present different types of files that contain all the information necessary to the plugin to work properly.

The files on which work are two: "*BAD.py*" and "*BAD_dialog_base.ui*".
The first one is a Python script and it will contain all the code in order "*to do things*". The script is edited and modified using Visual Studio Code as editor. The second, "*BAD_dialog_base.ui*", is the file that contains the elements and the style of the user interface: "*to be*". There are two ways in order to develop this script:

- writing from scratch all the elements and their properties (hand coding);
- using QtDesigner.

In the next paragraph it is explained why QtDesigner is used, and how it was used.

4.2. QtDesigner

The QtDesigner [51] is a free software that can be obtained automatically when QGIS is downloaded and installed. This software allows to read "*.ui*" files and to work on it in an easy and intuitive way.

QtDesigner offers the possibility to generate an User Interface (UI) without writing code but simply by using graphical tools. It is sufficient to use the Widget Box (generally tab on the left) in order to insert the needed graphical elements. The customization of the layout and the setting of the widgets properties, the developer can use the Property Editor (generally the window on the right)

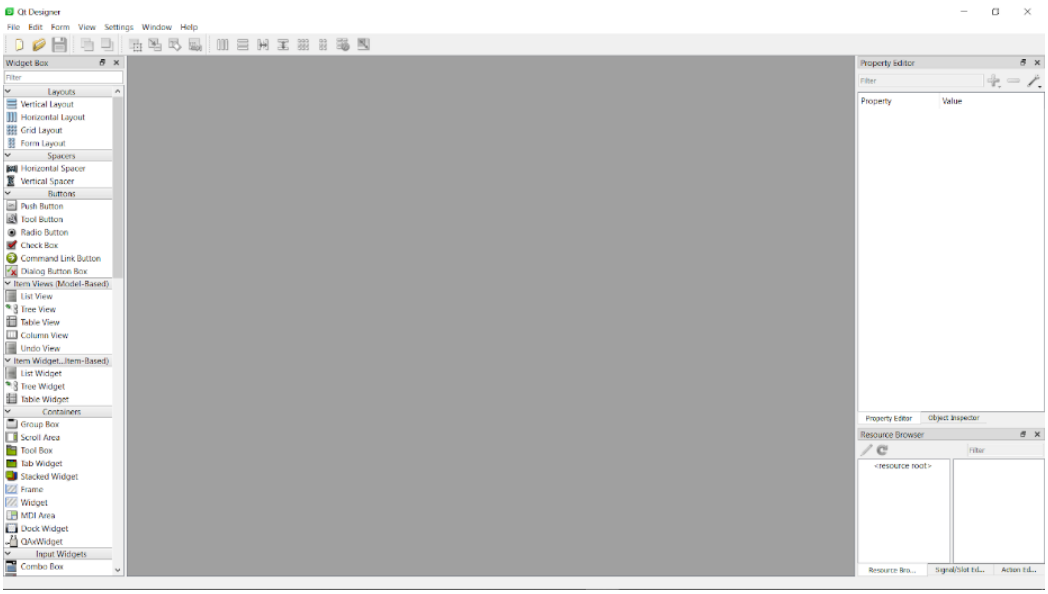


Figure 4.3 QtDesigner User Interface

After that the User Interface is graphically outlined, the QtDesigner generates the code.

The widgets that are used in the BAD plugin are the following:

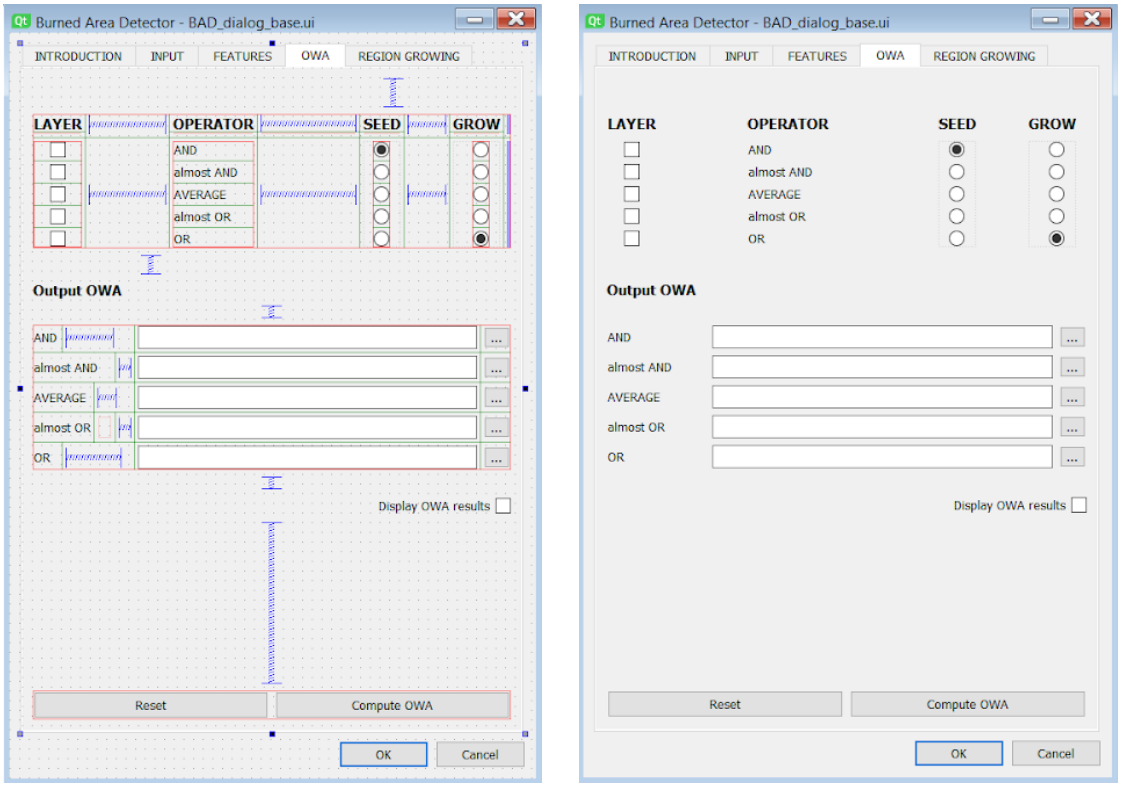
- dialog button box: this button is inserted by default by QtDesigner and it is generally composed of two buttons: "OK" button to execute actions and a "CANCEL" button to quit the dialog;
- combo box: it provides a means of presenting a list of options in a way that takes up the minimum amount of screen space;
- check box: it is a button that can be activated (checked) or deactivated (unchecked);

- radio button: similar to the check box, but to select “one of many” options that are present;
- spin box: it is a button to set an integer value as input. It is possible to fix the minimum and maximum value and also the step;
- double spin box: similar to the spin box but the value is a double (number with decimal part)
- push button: as suggested by the name, a button that can be clicked by the user;
- line edit: a space where the user can write whatever s/he wants. In BAD it will be used to write the path of the output file;
- tool button: a button that provides a quick-access to specific commands or options. In BAD it will be used to open folders in order to save an output without writing inside the line edit;
- label: static text that is used to write text by the developer.

Instead, the widgets used to modify, organize the layout are:

- tab widget: it provides a tab bar and a page area that is used to display pages related to each tab;
- tool box: a widget that displays a column of tabs one above the other, with the current item displayed below the current tab;
- group box: it is a container for widgets that organize buttons into groups, both logically and on screen;
- vertical/horizontal/grid layout: it specifies how to arrange the different elements with respect to other elements;
- horizontal/vertical spacer: it sets the distance (horizontal or vertical) between two widgets.

An example of a BAD’s tab is reported in Figure 4.4. On the left (a) it is possible to see the structure with almost all elements that are described previously, on the right (b) the final result: what an user will see.



(a) *(b)*

Figure 4.4 BAD's TAB element by element (a); The result BAD plugin TAB (b)

At this point it is clear to the reader how much Qt Designer is useful for creating a valid and effective User Interface, but some simple questions could immediately arise in his mind: how "*to do things*" when a button is clicked? How to exploit information that comes from the interface as input?

To answer these questions, it is necessary to introduce PyQt5, because QtDesigner supports the development of the interface, but actions are managed by PyQt5.

4.3. PyQt5

PyQt5 [52] is a comprehensive set of Python bindings for Qt version 5; using this tool, it is possible to give life to the plugin.

```
from .BAD_dialog import BADDialog
. . .
def run(self):
if self.first_start == True:
        self.first_start = False
        self.dlg = BADDialog()
```

Script 4.1 Connection between the Python code with the User Interface

These lines of code are crucial for the development of the plugin since they connect the Python code with the User Interface file and so have access to each widget in it.

Now, only those widgets that the user can interact with can be considered, and not the ones used to create the layout.

Another distinction must be done:

- widget to read information from: combo box, check box, radio button, spin box, double spin box, line edit;
- widgets that start doing things when triggered: dialog button box, push button, tool button.

From the first class it is possible to exploit information with the following lines of code:

```
self.dlg.comboBox_NAME.addItems([WHAT TO INSERT])
self.dlg.checkBox_NAME.isChecked()
self.dlg.radioButton_NAME.isChecked()
self.dlg.SpinBox_NAME.value()
self.dlg.doubleSpinBox_NAME.value()
self.dlg.lineEdit_NAME.text()
```

Script 4.2 Reading the information from UI provided by the user

In this way the information (provided by the user) is read and can be saved as a variable that can be used later in the code.

A different procedure is implemented for the second type of widgets: each of the buttons must be connected first of all; except for the dialog button which is connected

by default. Only after this operation is it possible to associate an action to a button ("self.ACTION").

At this point, if the user clicks on it, the button is triggered and executes the associated "ACTION".

```
self.dlg.toolButton_NAME.clicked.connect(self.ACTION)
self.dlg.pushButton_NAME.clicked.connect(self.ACTION)
```

Script 4.3 Button connection

Summarizing, at the beginning was described the PluginBuilder3 that allows to generate the basic structure, the skeleton of the plugin; then, the QtDesigner that gives the possibility to add elements and set the style; finally, it was described the PyQt5 that is responsible for the behavioral aspect of the widgets.

Next section describes in the details the development, the implementation and the functioning of the BAD plugin.

4.4. BAD implementation

The general scheme of the plugin is shown in Figure 4.1. Each step reported in that scheme represents a TAB in the BAD plugin: each of them receives inputs either from the user, or from the previous step or both and eventually can produce output. The algorithm implemented in the plugin works pixel by pixel and all the functions described in this chapter are assumed as applied for each pixel (cell of the raster/array).

This schema represents the Input/Output flow of BAD plugin:

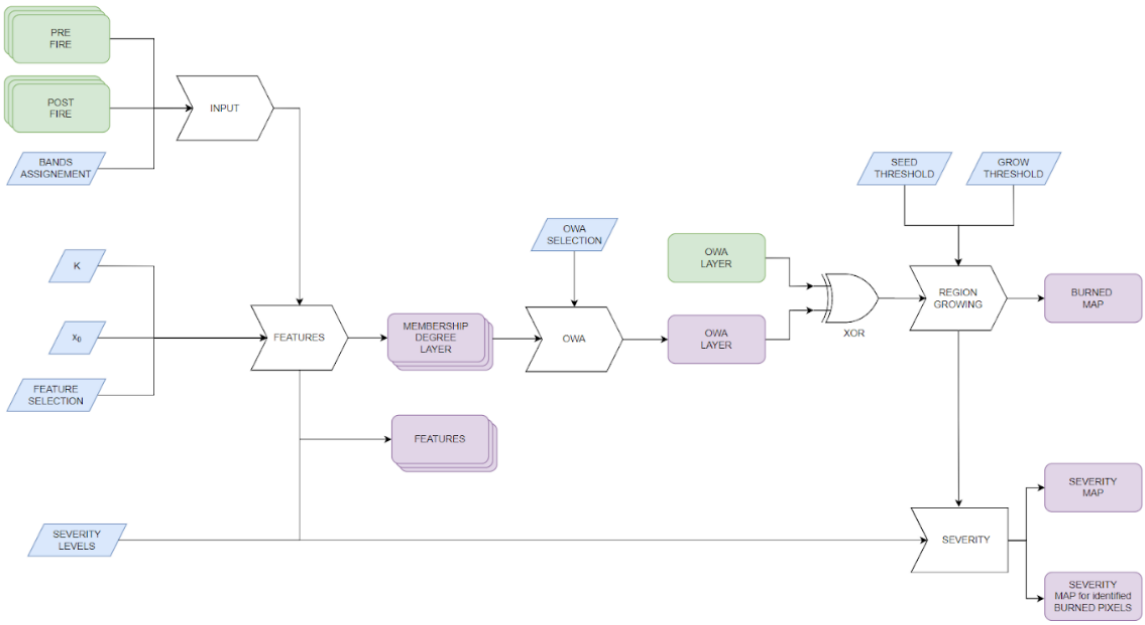


Figure 4.5 The structure of the plugin developed for QGIS with input/output flow; raster data types are represented by green boxes, inputs by the users are represented by blue boxes and purple boxes are outputs.

In the flowchart represented in Figure 4.5, green rectangles represent inputs in multi-band raster format (gridded data, as in the case of multi-bands satellite images), blue boxes are inputs like parameters or user choices (as in the case of threshold values), purple rectangles are output in single-band raster format data type.

To better understand how each page and its features work, each of them will be analyzed individually in the following paragraphs.

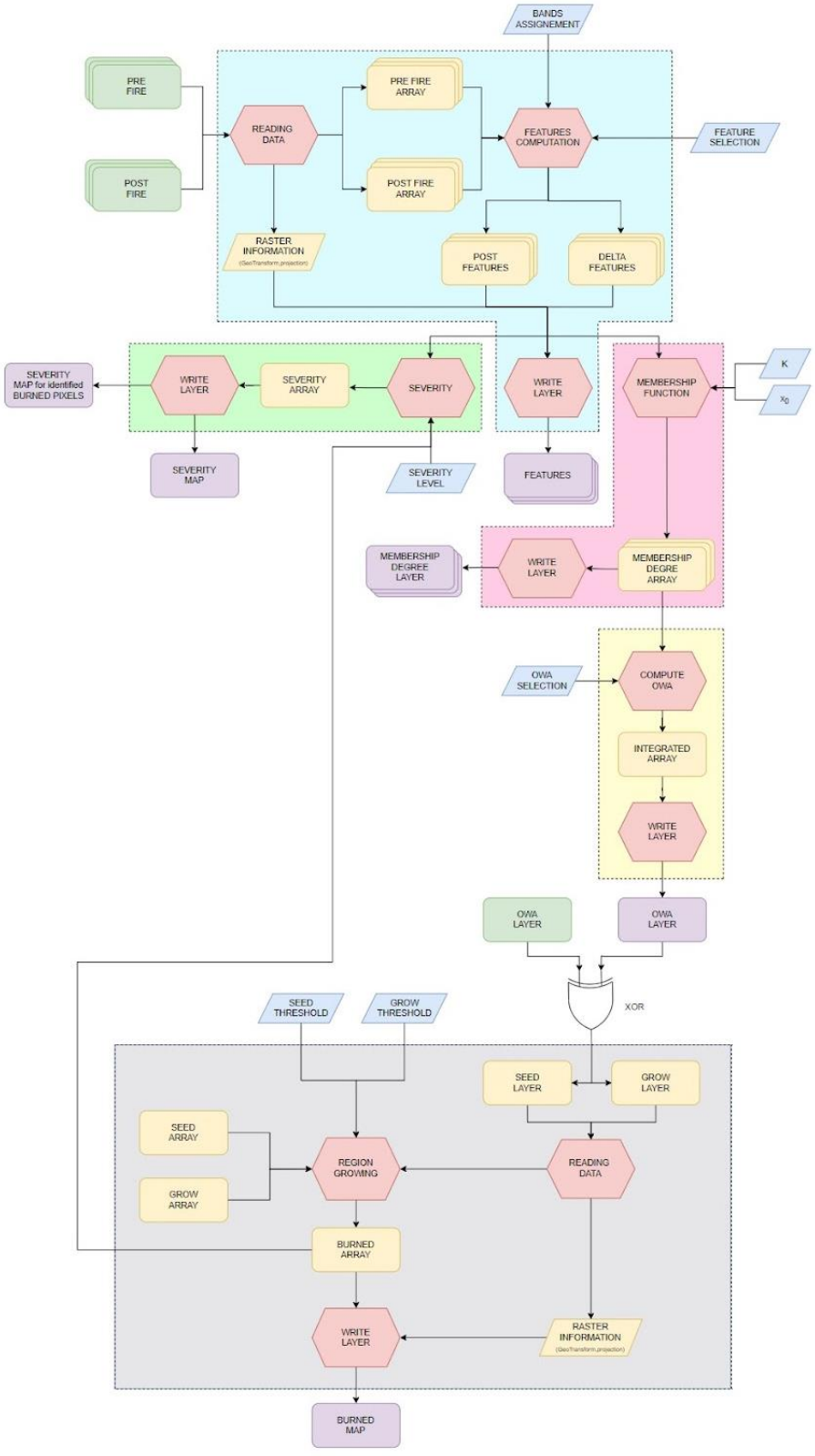


Figure 4.6 The detailed flow chart of BAD plugin. Each area represents what happens inside each TAB when a button is triggered: Features computation (cyan area); Membership Degree computation (pink area); OWA computation (yellow area); RG computation (gray area); Severity computation (green area).

4.4.1. TAB1 "INTRODUCTION"

This first TAB appears on the user's screen when the plugin is invoked in QGIS (clicking on the icon of the plugin).

The TAB briefly contains the instructions on how to use the plugin and the main information on the functioning of each tab.

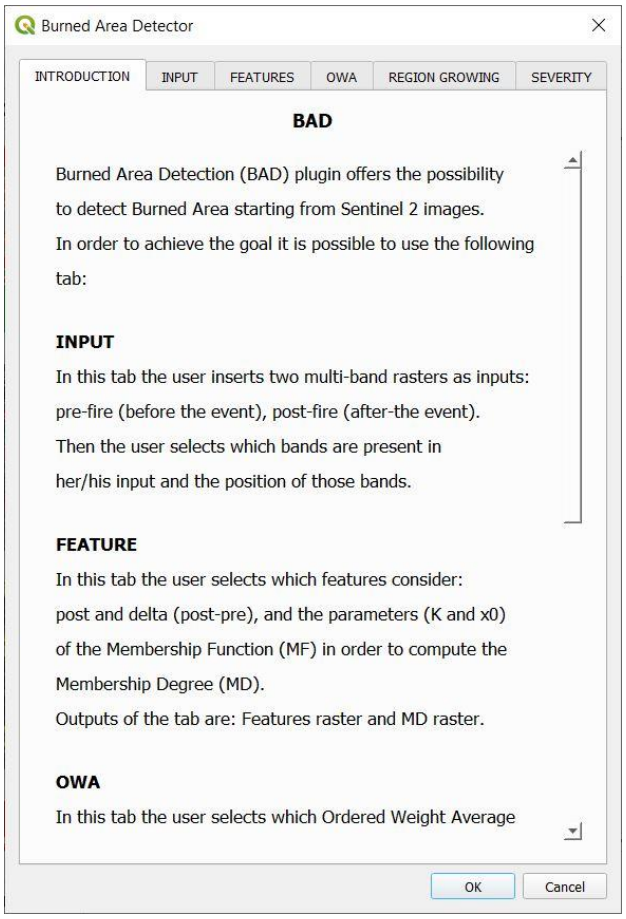
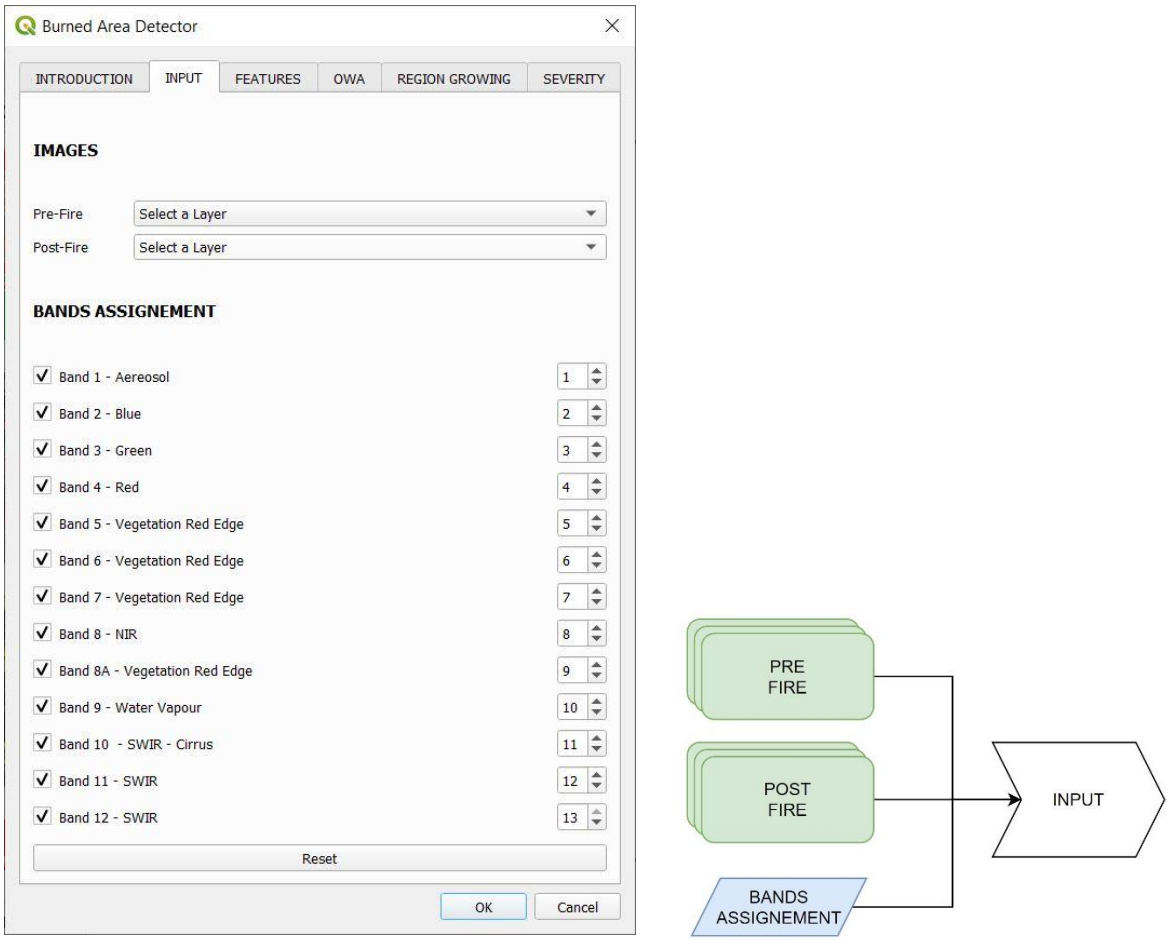


Figure 4.7 TAB1 "INTRODUCTION"

4.4.2. TAB2 "INPUT"

In this page the user begins interacting with the BAD plugin. The first task that is asked to her/him is to select the two input images: pre-fire image and post-fire image. These two inputs have to be multi-band raster layers and they must be opened in QGIS and visually present in the layer section.



(a) *(b)*

Figure 4.8 TAB2 "INPUT"(a); I/O for TAB2 (b)

One of the few constraints imposed by the plugin is that the two input images must have the same number of bands and the user must know which they are and how they have been assigned.

In fact, the fuzzy BA algorithm is designed to process multi temporal Sentinel-2 satellite multi spectral images that are composed of the same number of spectral bands (Chapter 3).

The input S-2 images could be available with a subset of the original bands and/or they can be ordered differently compared to the original multi-band Level 2A product. Hence, the section "BAND ASSIGNATION" allows the user to select which bands are present in her/his multi-band image (using the check boxes on the left) and set the numeric value indicating the position in the multi-layer raster (using the spin boxes on the right). In this section, naming of the layers follows S-2 naming convention (Table 1.01). For example, considering a multi-band S-2 satellite image, if,

for any reason, the user's raster file has Band8-NIR in the first layer, s/he must activate the Band8-NIR check box and set the spin box value to 1.

By default, all bands are checked (i.e., all bands are available in the input images) and values are set from 1 to 13, following the common structure of a S-2 raster image downloaded from the Copernicus Open Access Hub [14].

At the end of the tab is present a push button: "RESET". Clicking on it the default conditions in this page are restored.

4.4.2.1. Implementation

The UI is associate to a Python page that has the main role to read information provided by the user in TAB2. First of all, it is necessary to read the layers that the user uploads in QGIS and then display them in the combo boxes.
The following code implements these operations:

```
#Fetch the currently loaded layers
layers = QgsProject.instance().layerTreeRoot().children()

#Clear the contents of the comboBox from previous runs
self.dlg.comboBox_prefire.clear()
self.dlg.comboBox_postfire.clear()

#Populate the comboBox with names of all the loaded layers
self.dlg.comboBox_prefire.addItems(['Select a Layer'])
self.dlg.comboBox_postfire.addItems(['Select a Layer'])

self.dlg.comboBox_prefire.addItems([layer.name() for layer in layers])
self.dlg.comboBox_postfire.addItems([layer.name() for layer in layers])
```

Script 4.4 Reading the layers uploaded in QGIS and display them in the combo boxes inside the UI

The next step is to verify if a check box is checked or not, and if it is checked, read the corresponding band value assigned by the user and save it.

```
if self.dlg.checkBox_BAND#.isChecked():
        Band#=self.dlg.spinBox_BAND#.value()
else:
        Band#=np.nan
```

Script 4.5 Check boxes verification

This code must be repeated for all the thirteen bands. But to be more precise will be implemented and used later on in paragraph 4.4.3.1 during the Features computation that receives as input the image bands.

Finally, the "RESET" button is implemented. Below it is reported the generic implementation for "RESET", that is available in all TABs and the code is similar. Minor changes are applied to the general form, by changing the default values and the name of the widgets involved in resetting default conditions.

The first operation is to connect the button:

```
self.dlg.pushButton_RESET.clicked.connect(self.RESET_tab)
```

Script 4.6 RESET button connection

Then the "RESET_tab" function must be defined:

```
def RESET_tab(self):
        # Reset double spin and spin box
        self.dlg.doubleSpinBox_NAME.setValue(DEFAULT_VALUE)
        self.dlg.SpinBox_NAME.setValue(DEFAULT_VALUE)
        # Reset check box
        self.dlg.checkBox_RG_display.setCheckState(True)
        self.dlg.checkBox_RG_display.setCheckState(False)
        # Reset radio button
        self.dlg.radioButton_OWA_S_AND.setChecked(True)
        # Reset line edit
        self.dlg.lineEdit_RG_result.clear()
```

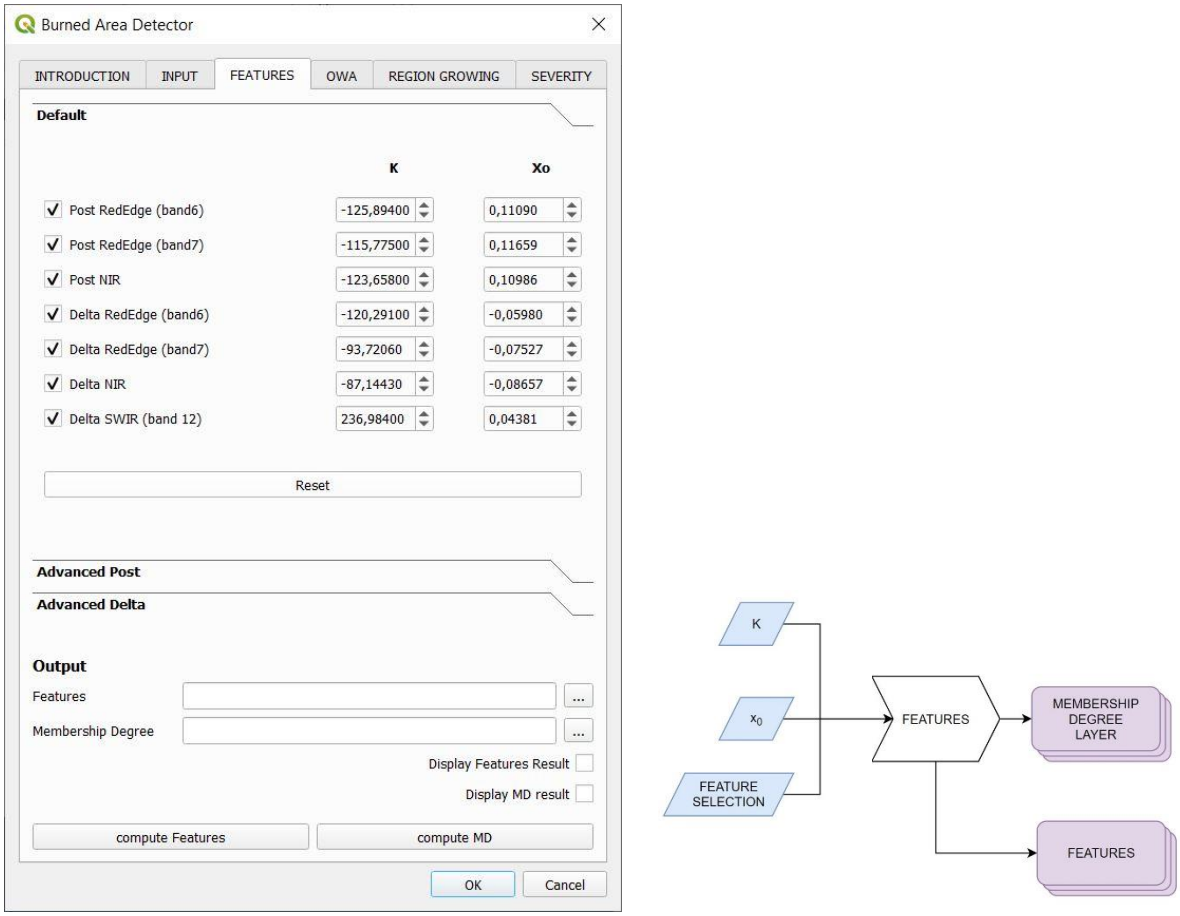
Script 4.7 RESET function declaration

4.4.3. TAB3 "FEATURES"

The goal of this tab is to collect information necessary to compute the Features and the Membership Degree (MD) of each pixel on.

The user selects on this TAB:

- the features that will be considered (post-fire reflectance bands and pre-post fire reflectance difference bands);
- the parameters (K and x0) that define the form of the Membership Function (MF).



(a) *(b)*

Figure 4.9 TAB3 "FEATURES" (a); I/O for TAB3 (b)

Also in this case, some features and some parameters are set by default. The default features are identified based on the literature as those bands that can better than others discriminate burned from unburned areas [(Sali et al., 2021) [2] (Table 3.1). The same for the proposed function parameters that best fit the frequency distribution of the features as proposed by Sali et al. (2021) [2] and trained for Mediterranean regions.
However, in order to offer the greatest flexibility, the BAD plugin offers the possibility to use different features (inside the page "ADVANCED POST" and "ADVANCED DELTA") and set the corresponding parameters for the MFs.
This allows the user to fit the BAD plugin to new scenarios and new regions. By default this advanced section is turned off and set to zero.

As it is now, however, BAD only implements a sigmoid shape function to derive membership degrees for each feature; future development could include the possibility for the user to change the shape of the function.

Obviously, if the user wants to use a specific feature related to a band, this band must be present in input multi-band raster and checked in TAB1.

4.4.3.1. Implementation

The implementation of this tab is based on the following operations:

- check the state of the check boxes;
- read the corresponding values;
- implement the "COMPUTE FEATURE" button;
- implement the "COMPUTE MD" (Membership Degree) button;
- save the output;
- display the result;
- implement the "RESET" button.

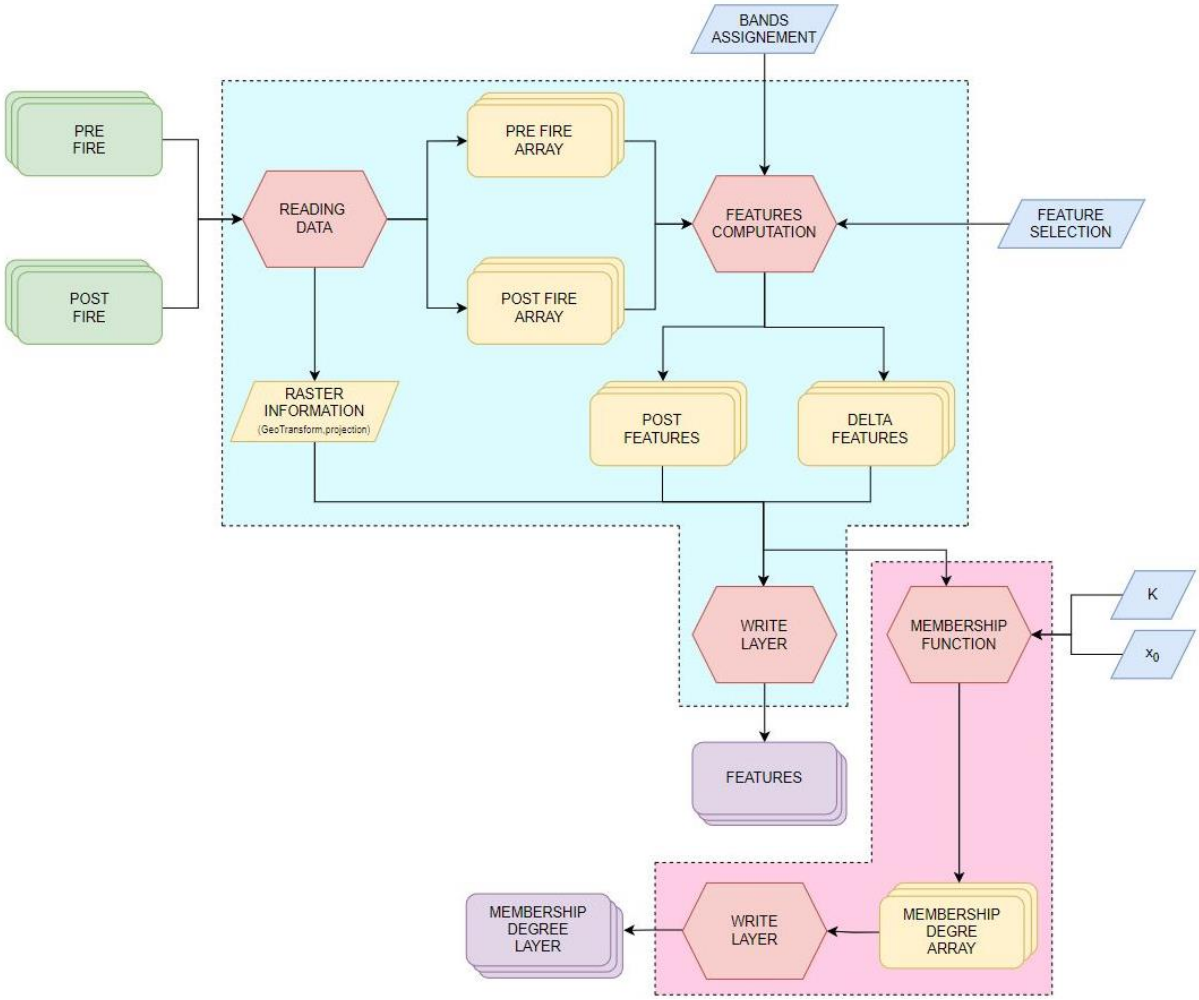


Figure 4.10 The flowchart of the steps of the Feature TAB implementation. In red are reported the function that "do things", instead in yellow are the products, the information that are

exploited, computed, used by BAD for the next steps. In green and blue the user's input (data and settings), finally in purple the output of BAD. Area in cyan represent the flow for COMPUTE FEATURE, area in pink for COMPUTE MD

How to control the state of a check box, and how to read values from a (double) spin box are already explained in paragraph 4.4.2.1. Also, the code for the "RESET" button is already described. Therefore, in this section, the remaining part of the code to develop is considered.

The first operation to do is to read (inside TAB1) which are the multi-band inputs (pre-fire and post-fire image) considered by the user and the selected bands. At this point the order of the bands is saved inside a list:

```
self.BandsList=[Band1,Band2,Band3,Band4,Band5,Band6,Band7,Band8,Band8A,Band9,
Band10,Band11,Band12]
```

Script 4.8 Band List declaration

The next step is to read the inputs, to do this the function "*ReadingData*" is called:

```
Data=ReadingData(self.Pre_path,self.Post_path)
```

Script 4.9 Calling ReadingData function

Input of the function are the paths of the input files.
The function exploits the *gdal.Open* function to open the raster inputs and to read the bands as matrices. From this point on BAD treats raster data as array data types. This will simplify the operations that will be done.

```
class ReadingData:
    def __init__(self,First_path,Second_path):

        First_raster = gdal.Open(First_path, gdal.GA_ReadOnly)
        Second_raster = gdal.Open(Second_path, gdal.GA_ReadOnly)
        # Note GetRasterBand() takes band no. starting from 1 not 0
        First_matrix=First_raster.ReadAsArray()
        Second_matrix=Second_raster.ReadAsArray()
        self.FirstMatrix=First_matrix.astype(float)
        self.SecondMatrix=Second_matrix.astype(float)

        self.Bands=First_raster.RasterCount
        self.gt = First_raster.GetGeoTransform()
        self.proj= First_raster.GetProjection()
```

Script 4.10 ReadingData function declaration

The "*ReadingData*" function gives the following outputs:

- FirstMatrix (pre-fire), SecondMatrix (post-fire);
- the number of bands (integer);
- geo transform: six coefficients that represent the affine transformation implemented by the "GDALDataset::GetGeoTransform()" function which map image coordinate space (row, column) to the georeferenced coordinate space (array) (projected or geographic coordinates);
- map projection (string).

At this point it is possible to compute the Delta Feature matrix:

```
self.DeltaMatrix=self.PostMatrix-self.PreMatrix
```

Script 4.11 DeltaMatrix computation

Before clicking on "COMPUTE FEATURE" the user has selected the desired features. Therefore only those features must be considered:

```
FinalFeatureList=[]
NameBandsList=[]
if self.dlg.checkBox_FEATURE.isChecked():
        layer=self.Matrix[Band#-1]
        FinalFeatureList.append(layer)
        NameBandsList.append('NAME_FEATURE')
```

Script 4.12 Collecting the feature selected by the user

The code inside if condition must be repeated for all the features. In this way only the selected features are present inside "*FinalFeatureList*".

Now, the output is saved. This task is carried out by a function specifically developed for this plugin: "*WriteLayer*()".

```
self.Feature_result= WriteLayer(path_index,path,FinalFeatureMatix,NameBandsList,Nband,
Xsize,Ysize,filename,self.GeoTrans,self.proj)
```

Script 4.13 Calling WriteLayer function

This procedure is quite similar to the "*ReadingData*". In fact, passing the information that are computed, and some that are read from the input it is possible to write a "*.tif*" file thanks to gdal library.

```
class WriteLayer:
   def __init__(self,index,path,Matrix,NameBands,Nband,Xsize,Ysize,filename,gt,proj):
      if index==0:
         array_path=path.split('/')
         array_path[-1]=filename
         splitter="/"
         self.output_path=splitter.join(array_path)
      else:
         self.output_path=path
      driver = gdal.GetDriverByName("GTiff")
      driver.Register()
      outds = driver.Create(
      self.output_path,xsize=Xsize,ysize=Ysize,bands=Nband,eType=gdal.GDT_Float32)
      outds.SetGeoTransform(gt)
      outds.SetProjection(proj)
      if Nband==1:
         outband = outds.GetRasterBand(Nband)
         outband.WriteArray(Matrix)
         outband.SetDescription(NameBands)
         outband.FlushCache()
         outband= None
      else:
         for i in range(Nband):
            outband = outds.GetRasterBand(i+1)
            outband.WriteArray(Matrix[i])
            outband.SetDescription(NameBands[i])
            outband.FlushCache()
            outband= None
      outds = None
```

Script 4.14 WriteLayer function declaration

There are two scenarios in order to save the output:

- the user specifies the path;
- the user does not specify the path.

In the first case the path is read and passed to the function. If this does not happen the output is automatically saved in the folder where the multi-band pre-fire input is stored.

```
Feature_outputfile = self.dlg.lineEdit_Feature.text()
      if not Feature_outputfile:
```

```
            path_index = 0
            path = self.Pre_path
            filename = "Feature_result.tif"
        else:
            path_index = 1
            path = Feature_outputfile
            filename="null"
```

Script 4.15 Path selection for the generated file

In the same TAB it is possible to also click the "COMPUTE MD" button. Clicking on it, the Membership Degree (MD) for each Feature is computed.

In order to compute MD it is necessary to read the "K" and "x0" values set by the user if checked.

```
if self.dlg.checkBox_FEATURE.isChecked():
            K=self.dlg.doubleSpinBox_FEATURE_K.value()
            x=self.dlg.doubleSpinBox_FEATURE_x.value()
            layer=self.FEATURE_Matrix[INDEX_OF_BAND]
            MD_FEATURE = MembershipFunction(layer,K,x).MD
            FinalBandList.append(MD_FEATURE)
            NameBandsList.append('MD_FEATURE')
```

Script 4.16 Reading information in order to compute the MD for the selected Features

Calling the developed function, "*MembershipFunction*", the Formula 3.1 about Membership Function Sigmoid is applied:

```
class MembershipFunction:
    def __init__(self,Matrix,K,x):
        self.MD=1/(1+np.exp(-K*(Matrix/10000-x)))
```

Script 4.17 MembershipFunction declaration

In the end, when MD is computed for each Feature, the output is saved. The process is similar to what was described for the Features output before.

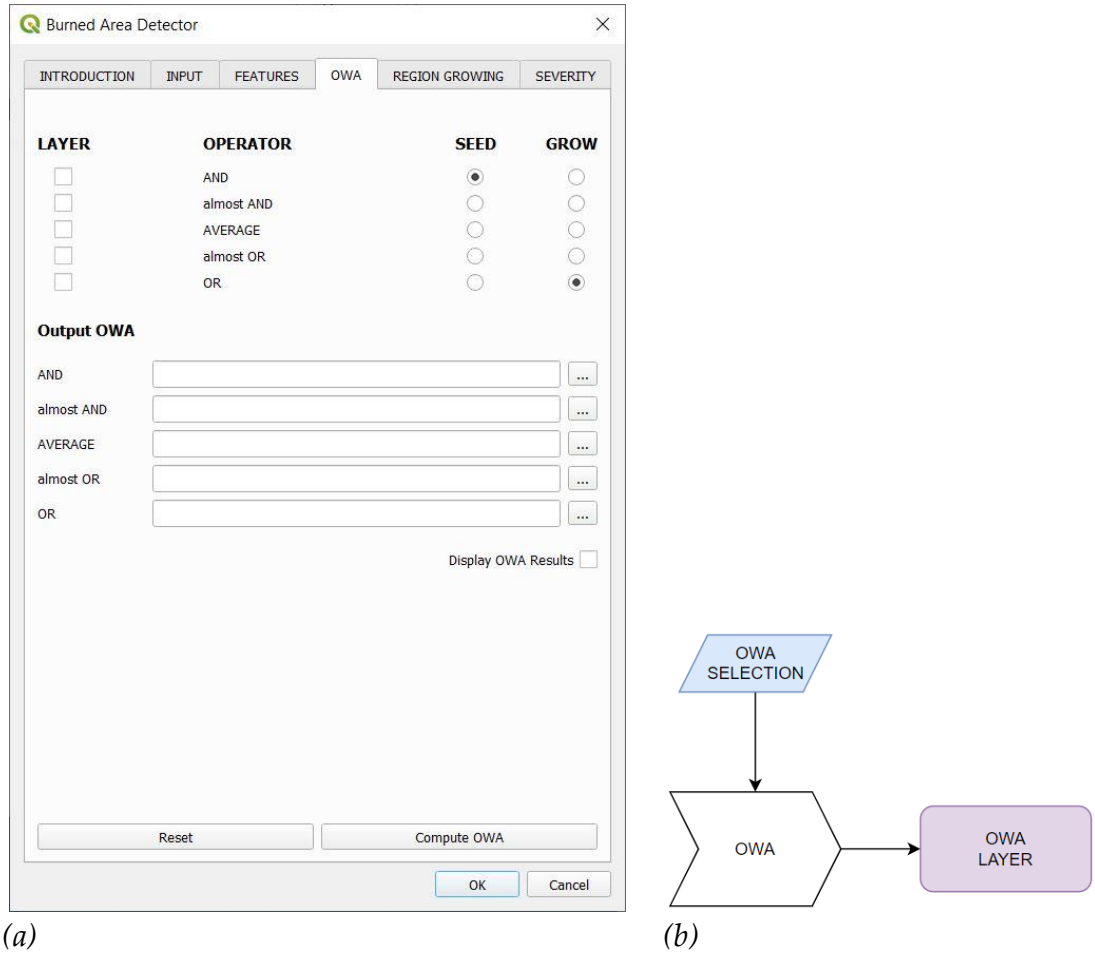
The outputs (Features and MD raster) can be displayed on the screen if the corresponding check box is activated.

```
if self.dlg.checkBox_NAME_display.isChecked():
            iface.addRasterLayer(OUTPUT_PATH, "NAME_TO_DISPLAY")
```

Script 4.18 Displaying the result

4.4.4. TAB4 "OWA"

In the fourth tab the user selects the OWA operator that she/he wants to use in order to generate the integrated layer from the selected input features. A set of OWAs are available (OWA_AND, OWA_AlmostAND, OWA_AVERAGE, OWA_AlmostOR, OWA_OR) based on the fuzzy BA algorithm implemented in this thesis work.



(a) *(b)*

Figure 4.11 TAB4 "OWA" (a); I/O for TAB4 (b)

The tab appearance is shown in Figure 4.11 (*a*). In the left column, under the label LAYER, there is the possibility to select the desired OWAs by checking a single one to all available layers. The layers that are checked will be computed and saved; the user can specify the path for the output in the edit line below ("Output OWA"), if output path is not specified, the BAD plugin saves the output layers as raster files in the same folder as the input files (i.e., the folder where there is the pre-fire image).

Moreover, in this tab the user can select which OWA will be used as seed and grow layers in the next step: Region Growing.

The choice is done by selecting one radio button under the SEED column, and one under GROW; default values are OWA_AND and OWA_OR for the seed and grow layers, respectively.

The presence of the two widgets (check box and radio button) keeps separated the left column (all elements can be selected) and the two right ones (only one element per column can be selected). Furthermore, each column on the right (seed and grow) is designed inside a frame so that two choices can be activated to select two layers from all those present.

4.4.4.1. Implementation

The implementation of this page is based on these operations:

- check the state of the checkboxes and the radio buttons;
- implement the COMPUTE OWA button;
- save the result;
- eventually display the result in the QGIS interface.

The implementation of the OWA operators follows the steps outlined in Chapter 3.

As shown in Figure 4.12, the input of this phase is the MD array computed in the previous step ("COMPUTE MD") and the OWA operator selected by the user.

```
if self.dlg.checkBox_OWA.isChecked() or self.dlg.radioButton_OWA_SEED.isChecked() or
\ self.dlg.radioButton_OWA_GROW.isChecked():
OWA_index=#
OWA=OrderedWeigthAverage(OWA_index,FinalBandMatix)
```

Script 4.19 Calling the OWA function selected by the user

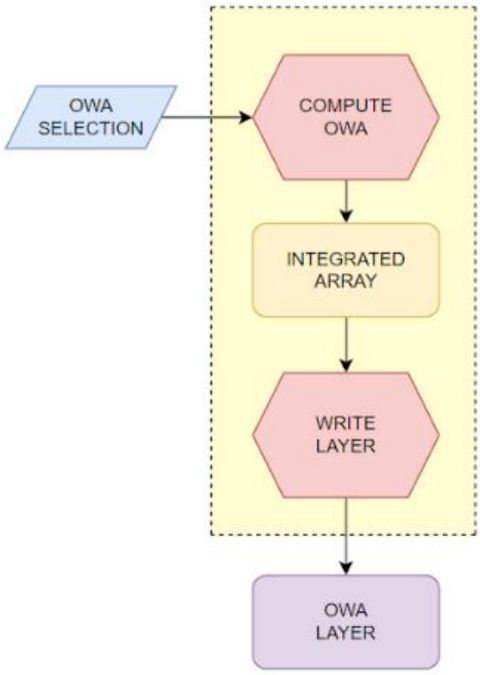


Figure 4.12 The flowchart of the steps of the OWA TAB implementation. In red are reported the function that "do things", instead yellow boxes are the products, the information that are exploited, computed, used by the BAD for the next steps. In blue the user's setting input finally in purple the output of the BAD plugin.

OWA_index is a numeric parameter used in the plugin to identify the OWA operators selected by the user:

- OWA_index = 1 -> AND;
- OWA_index = 2 -> almostAND;
- OWA_index = 3 -> AVERAGE;
- OWA_index = 4 -> almostOR;
- OWA_index = 5 -> OR.

The developed function code is reported below. It must be replicated for each of the five OWA.

```
class OrderedWeigthAverage:
def __init__(self,index,FinalBandMatix):
        Row=FinalBandMatix.shape[1]
        Column=FinalBandMatix.shape[2]
        self.Integrated_matrix=np.empty([Row,Column])
```

```
if index==#:
    self.filename= "OWA_***.tif"
    for i in range (Row):
        for j in range (Column):
            vector=FinalBandMatix[:,i,j]
            if i==0 and j==0:
                value=###(vector)
                self.Integrated_matrix[i,j]=value
        if index==#:
                ...
```

Script 4.20 OWA function declaration

As explained in details in chapter 3 (Table 3.2), the OWA value is computed in the following way:

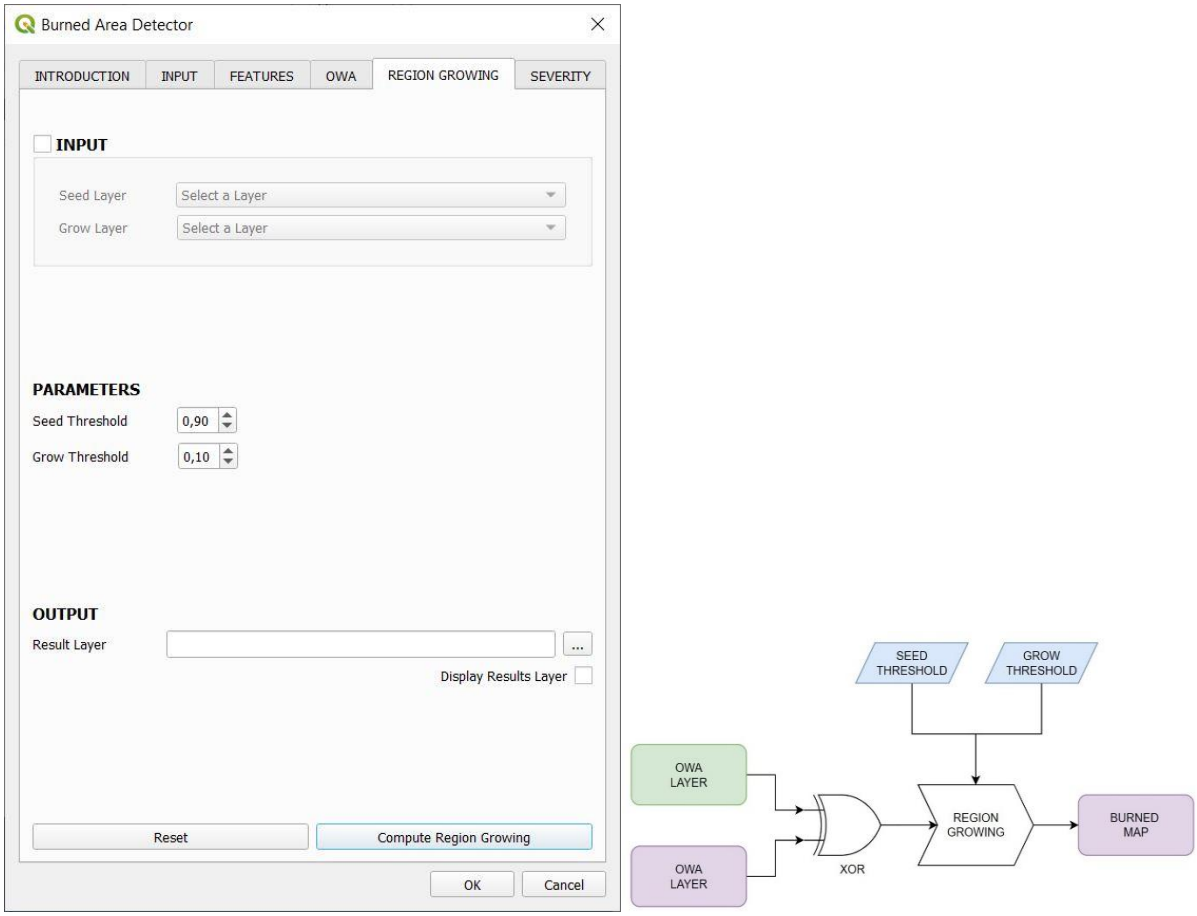
- OWA_AND -> minimum value;
- OWA_almostAND -> mean between the minimum value and the second smallest;
- OWA_AVERAGE -> mean of the values;
- OWA_almostOR -> mean between the maximum value and the second biggest;
- OWA_OR -> maximum value.

Once OWA computation is executed, it is possible to write the output layer and to show the result in QGIS interface (Layer section), if the display checkbox is activated by the user.

These tasks are already well explained in the previous paragraph.

4.4.5. TAB5 "REGION GROWING"

This TAB either could be the following step for the user or could be the starting point.



(a) *(b)*

Figure 4.11 TAB5 "REGION GROWING" (a); I/O for TAB5 (b)

The user when opens this page she/he has the two layers: seed and grow layer. These two layers can be generated from the previous step of BAD or are computed with other methods. If they are derived from other programs, the user has to activate the "INPUT" check box and select them. Otherwise this procedure is not necessary.

The only condition that must be satisfied is: the input for the RG algorithm must be layer with only one band and the value of each pixel must have a value between 0 and 1.

After that, the user sets the two thresholds that will be used during the RG algorithm and, if s/he wants, the path where to save the output.

4.4.5.1. Implementation

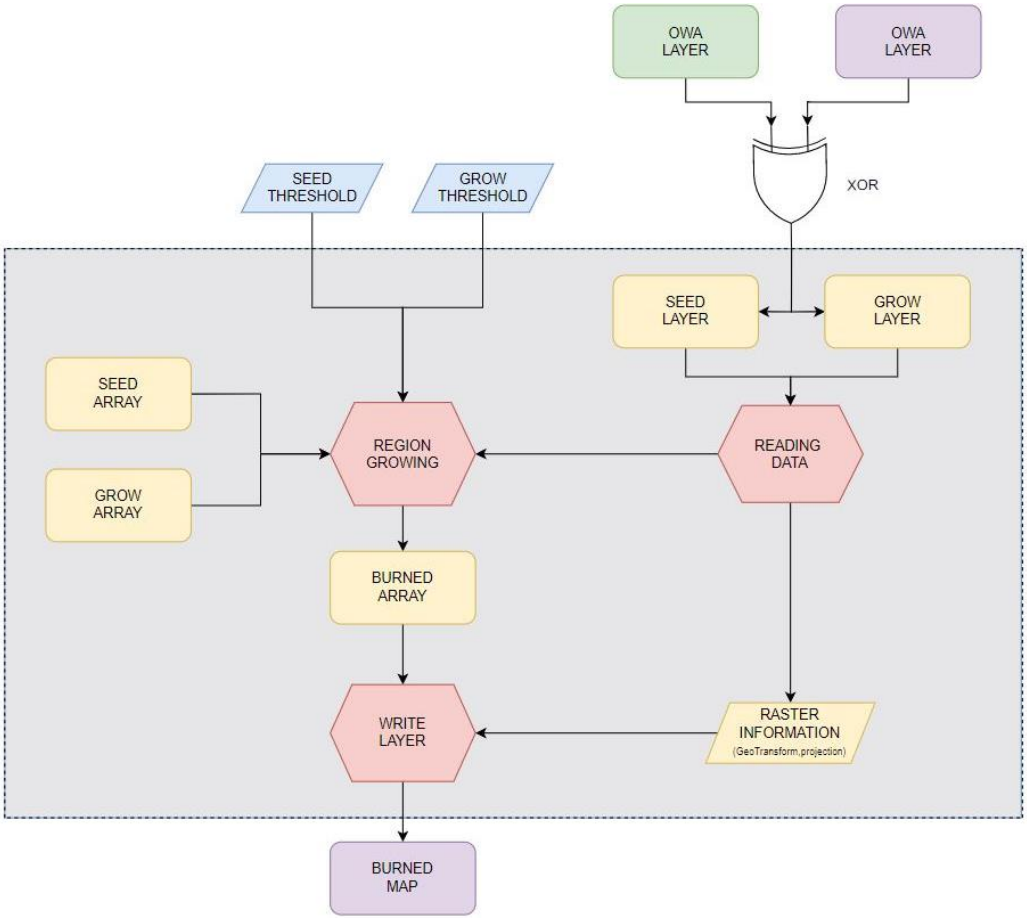


Figure 4.12 The flowchart of the steps of the RG TAB implementation.
In red are reported the function that "do things", instead in yellow are the products,
the information that are exploited, computed, used by BAD for the next steps.
In blue the user's setting input finally in purple the output of BAD plugin.

For this TAB will be necessary to spend more time in the discussion due to the fact that a RG algorithm was already present, but was implemented in Harris IDL language, that is not an open free software.

So, it was implemented a new version, always following the theory well explained in chapter 3, using Python that is free and accessible to everyone.

The RG algorithm is an iterative algorithm. Each iteration is a growing process that considers one pixel at a time to select seeds from those matrix elements in the seed layer that are equal to 1; afterwards, the algorithm checks the eight neighbor pixels of each seed to verify if the growing condition is satisfied (OWAgrow greater than a threshold). Those neighboring pixels that satisfy the condition are selected and become new seeds. This procedure is repeated until no new seeds are found or the border of the matrix is reached.

This process is iterative and for this reason, can be computationally demanding.
In order to develop the RG code was implemented the following script: scrolling the matrix pixel by pixel (Pixel by Pixel method).

4.4.5.2. Pixel by Pixel method

This approach is based on scrolling through the OWAseed matrix thanks to two "*for*" cycles: seeds are identified as elements with value 1. At this point, when a seed is found, its eight neighbors are analyzed. Some conditions are added in order to avoid errors during the execution of the code:

- an element in the first column has no neighbors on the left;
- an element in the last column has no neighbors on the right;
- an element in the first row has no neighbors above;
- an element in the last row has no neighbor below.

For the selected neighbors' pixels/elements, if they are not already seeds (OWAseed = 0), values inside the grow matrix are considered.
If the condition (OWAgrow = 1) is satisfied, the element is added to the seed matrix.
When each element of the seed matrix has been tested, the entire procedure is repeated with the new seed matrix, this is done until no new seeds are added between two consecutive cycles.

```
class RegionGrowing:
   def __init__(self,Seed_threshold,Grow_threshold,Seed_matrix,Grow_matrix):

      Seed_binary = np.where(Seed_matrix<Seed_threshold,0,1)
      Grow_binary = np.where(Grow_matrix<Grow_threshold,0,1)
      self.Raster=np.array([Seed_binary,Grow_binary])
      RasterPP=self.Raster.copy()

      sizeRow=RasterPP[0].shape[0]
      sizeColumn=RasterPP[0].shape[1]
      M=0
      N=np.count_nonzero(RasterPP[0])

      while N!=M:
         M=N
         for j in range(sizeRow):
            for k in range(sizeColumn):
```

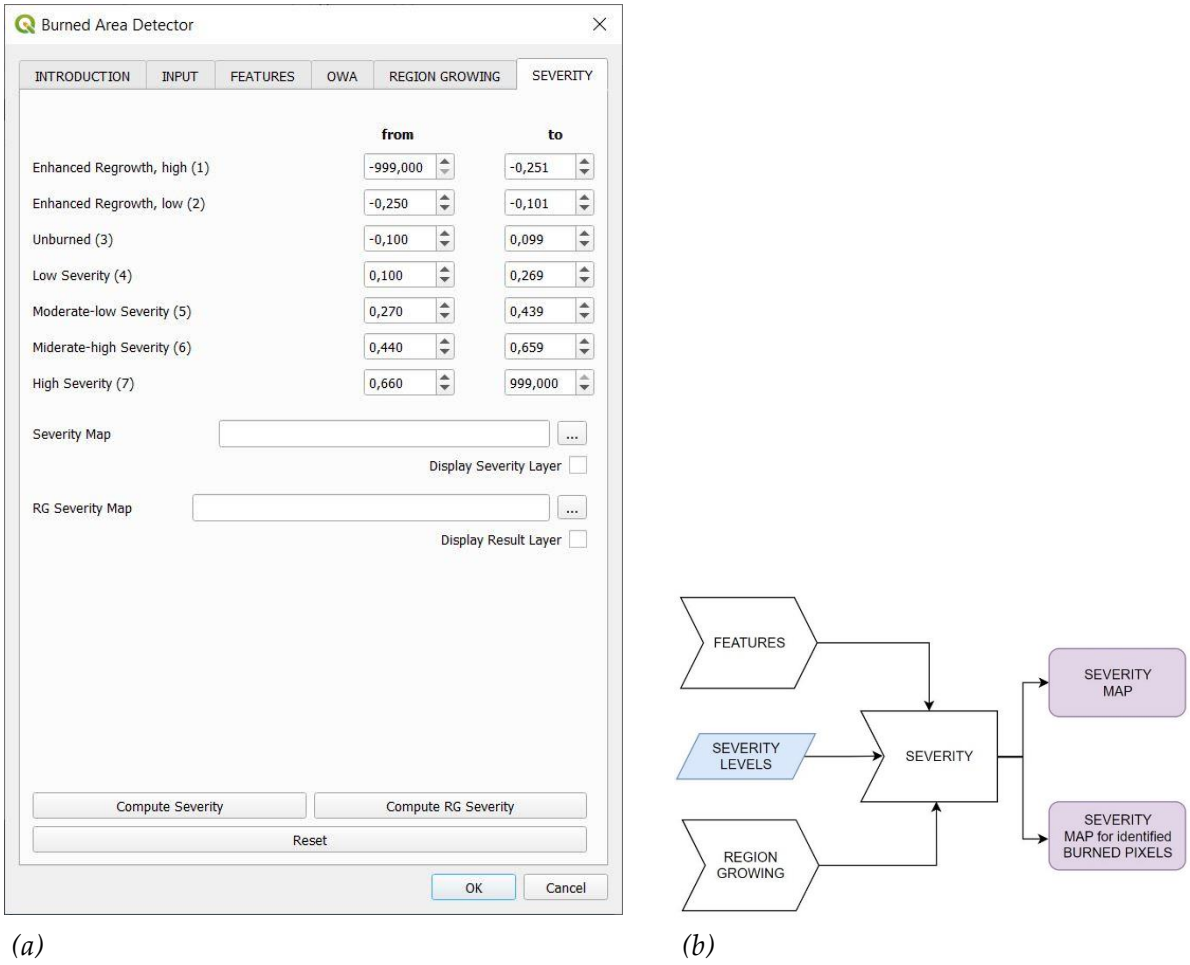
```
#insertion of the neighbors that are not considered yet

        if RasterPP[0,j,k]==1:

            # neighbor examination
            # above-left
            if j>0 and k>0 and RasterPP[0,j-1,k-1]==0 and RasterPP[1,j-1,k-1]==1:
                RasterPP[0,j-1,k-1]=1

            # above
            if j>0 and RasterPP[0,j-1,k]==0 and RasterPP[1,j-1,k]==1:
                RasterPP[0,j-1,k]=1

            #above-right
            if j>0 and k<=sizeColumn-2 and RasterPP[0,j-1,k+1]==0 and
            RasterPP[1,j-1,k+1]==1:
                RasterPP[0,j-1,k+1]=1

            #left
            if k>0 and RasterPP[0,j,k-1]==0 and RasterPP[1,j,k-1]==1:
                RasterPP[0,j,k-1]=1

            #right
            if k<=sizeColumn-2 and RasterPP[0,j,k+1]==0 and RasterPP[1,j,k+1]==1:
                RasterPP[0,j,k+1]=1

            #below-left
            if j<=sizeRow-2 and k>0 and RasterPP[0,j+1,k-1]==0 and
            RasterPP[1,j+1,k-1]==1:
                RasterPP[0,j+1,k-1]=1

            #below
            if j<=sizeRow-2 and RasterPP[0,j+1,k]==0 and RasterPP[1,j+1,k]==1:
                RasterPP[0,j+1,k]=1

            #below-right
            if j<=sizeRow-2 and k<=sizeColumn-2 and RasterPP[0,j+1,k+1]==0 and
            RasterPP[1,j+1,k+1]==1:
                RasterPP[0,j+1,k+1]=1

    #condition that blocks the while
    N=np.count_nonzero(RasterPP[0])
    self.Result_matrix=RasterPP[0]
```

Script 4.21 RegionGrowing function declaration (Pixel by Pixel method)

4.4.6. TAB6 "SEVERITY"

In this last TAB the user can produce two possible outputs:

- the Severity Map (paragraph 2.3);
- the combination between the Severity and the RG output: Severity Map for identified burned pixels. The information that comes from this last product is the severity only for those pixels that are classified as "Burned" by the RG algorithm.



(a) *(b)*

Figure 4.13 TAB6 "SEVERITY" (a); I/O for TAB6 (b)

Both outputs are multi-class raster: each pixel presents a single value that describes at which class it belongs.
For the Severity Map the classes are eight: from label 1 to 7, as reported in Table 2.1, in order to express the severity of the wildfire event computed for each pixel (paragraph 2.3), the last class (label value equal to 999) represent the masked pixel;

For the second output the classes are the same of the Severity Map except for these two differences:

- due to the fact that this output is the combination of two classification outputs (RG algorithm and dNBR algorithm) an additional class must be considered. The class that must be added come from RG classification, and it is the "Unburned" pixel's class (label value equal to 0);
- the label value for masked pixel is equal to 1000.

The Severity computation is the first step. Therefore, the user has to set the interval for the severity levels (Table 2.1) using the double spin boxes. Then the user can produce the combined map: Severity Map for identified burned pixels.

Finally the user can decide where to save the output files and if to display them on the screen or not.

4.4.6.1. Implementation

In order to implement this TAB it is necessary to have the following input data, as shown in Figure 4.14:

- the input bands (Band8 and Band12) acquired from pre fire image and post fire image.

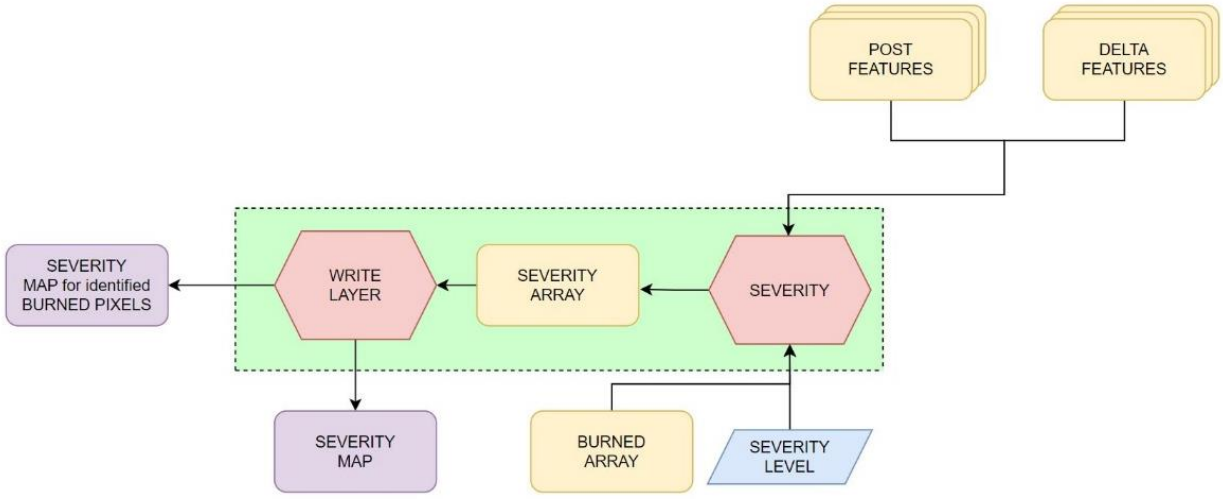


Figure 4.14. The flowchart of the steps of the Severity TAB implementation. In red are reported the function that "do things", instead in yellow are the products, the information that are exploited, computed, used by BAD for the next steps. In blue the user's setting input finally in purple the output of BAD plugin

These lines of codes recall part of the multi-band input (pre-fire, post-fire) already read during the feature computation;

```
def ComputeSeverity(self):
    NIR_pre=self.PreMatrix[self.BandsList[7]-1]
    NIR_post=self.PostMatrix[self.BandsList[7]-1]
    SWIR_pre=self.PreMatrix[self.BandsList[12]-1]
    SWIR_post=self.PostMatrix[self.BandsList[12]-1]
```

Script 4.22 NIR and SWIR acquisition

- the severity levels, in order to read the levels it is sufficient to read the double spin boxes:

```
level= self.dlg.doubleSpinBox_Severity_level_value()
```

Script 4.23 Reading the Severity level

and save it inside lists, one for the lower bound of the range and one for the upper bound:

```
LowerLevelList.append(lower_bound)
UpperLevelList.append(upper_bound)
```

Script 4.24 Severity level range declaration: upper and lower bound

With these bands it is computed the index NBR for the pre-fire and the post-fire images (Formula 2.1). At this point, with preNBR and postNBR, and the severity level range, it is possible to produce the Severity Map using the Formula 2.2:

$$dNBR=pre\ NBR-post\ NBR$$

and applying the classification (Table 2.1).

```
Severity_result=Classification(Delta_NBR,LowerLevelList,UpperLevelList)
```

Script 4.25 Calling Classification function

When the "*Classification*" function is called seven matrices are generated, one for each severity interval. Each of them is generated in the following way:

```
Matrix_Level_i =
np.where((Matrix>=LowerLevelList[i-1])&(Matrix<=UpperLevelList[i-1]),i,0)
```

Script 4.26 Matrix Level declaration for Severity class "i"

Thanks to the "*np.where*" function each element of the matrix is considered and, if its value falls within the interval, the class of that element is set to i ($i \in [1, 7]$, Table 2.1), otherwise to 0.

This procedure is repeated for each class. At the end the matrices are summed in order to obtain a single classification (this final matrix is the output of the classification function).

This approach can be used due to the fact that the value of the element/pixel for sure falls only in a range and so belongs to a single class, conditions are in fact exclusive. For example, if the element belongs to the "*Unburned*" class, its class values in each of the matrices would be: [0,0,3,0,0,0,0]. Therefore, summing these values the result will be equal to 3, that represents the "*Unburned*" class.

```
class Classification:
    def __init__(self,Matrix,LowerLevelList,UpperLevelList):
        Matrix=np.round(Matrix,3)
        M_ERH  = np.where((Matrix>=LowerLevelList[0])&(Matrix<=UpperLevelList[0]),1,0)
        M_ERL  = np.where((Matrix>=LowerLevelList[1])&(Matrix<=UpperLevelList[1]),2,0)
        M_U = np.where((Matrix>=LowerLevelList[2])&(Matrix<=UpperLevelList[2]),3,0)
        M_L = np.where((Matrix>=LowerLevelList[3])&(Matrix<=UpperLevelList[3]),4,0)
        M_ML = np.where((Matrix>=LowerLevelList[4])&(Matrix<=UpperLevelList[4]),5,0)
        M_MH   = np.where((Matrix>=LowerLevelList[5])&(Matrix<=UpperLevelList[5]),6,0)
        M_H = np.where((Matrix>=LowerLevelList[6])&(Matrix<=UpperLevelList[6]),7,0)

        self.Final_Matrix = M_ERH + M_ERL + M_U + M_L + M_ML + M_MH + M_H
```

Script 4.27 Classification function declaration

At this point, with the output matrix, a raster is generated using the "*WriteLayer*" function, and eventually displayed on the screen.

In order to produce the second output (the combination between the Severity and the RG maps) it is sufficient to multiply the two layers. The output that comes from this operation is a raster layer, in which each pixel can assume one of the possible eight value:

- 0, if the pixel in the RG output is classified as "*Unburned*" (0);
- from 1 to 7, if the pixel in the RG output is classified as "*Burned*" (1) and the value depends on the Severity assigned to the pixel.

After the operation, the function "*WriteLayer*" is invoked in order to produce and save the raster, and eventually displayed it on the screen.

5 TEST AND RESULT

The BAD plugin developed in this thesis work was first tested over a burned area in Portugal. This was the prototyping phase of the development. Successively, the plugin is used to analyze some real scenarios affected by wildfire in order to make the validation. Two types of analysis were carried out during the test phase to assess both processing time and the accuracy of the burned area map obtained as output. The processing time was measured to quantify the effort and the resources needed for deriving the burned area map from the input Sentinel-2 pre- and post-fire images. In all test cases, the fire perimeters provided by the Copernicus EMS-Mapping service were used as reference dataset for the comparison of BAD mapping output and the assessment of mapping accuracy.

The first case study used during the development phase was the scenario of the Pedrógão Grande fire that occurred in Portugal in 2017.

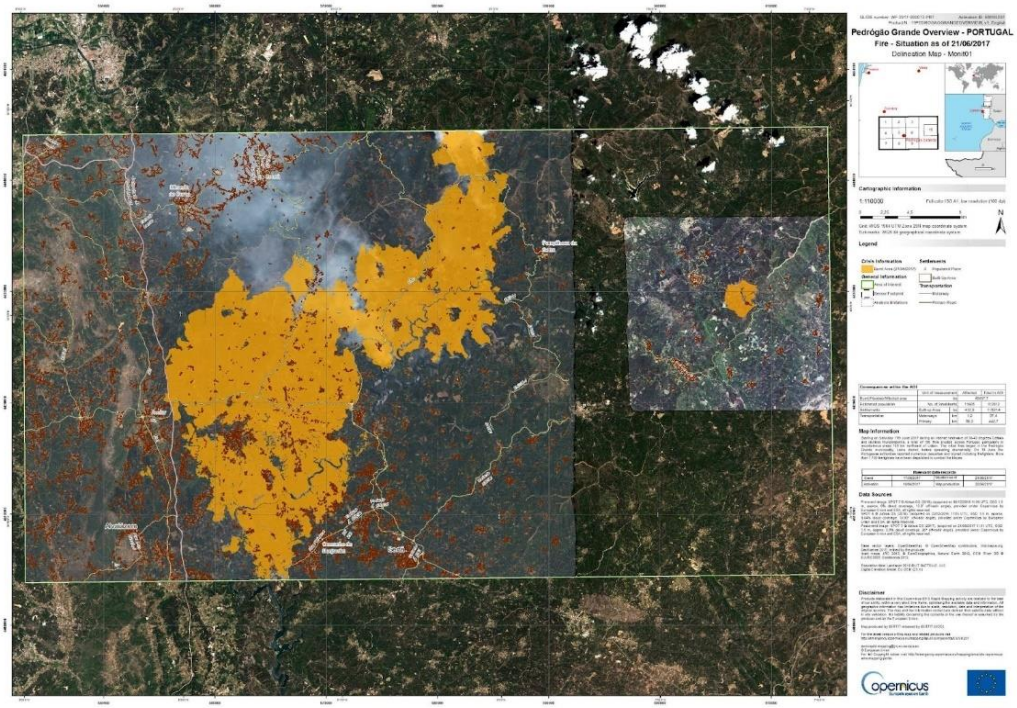


Figure 5.1 EMSR207 delineation map for Portugal case study

This fire broke out on June 17th, 2017, with multiple ignition sources that led to a catastrophic event due to severe drought conditions.
The severe conditions of fire burning required the activation of the Copernicus EMS-Mapping service that was registered as EMRS207 [53].

In this case, the EMS fire perimeters were derived from very high resolution SPOT satellite images (pre-fire: 09-22/12/2016 and post-fire 23/06/2017).

Other two test cases were identified in the most recent fire events.
The first case study is the wildfire in the Valencian Community, Spain, which occurred in early August 2022. In particular, a wildfire broke out in the Bejís forest, in the province of Castellón, identified by the Copernicus EMS-Mapping service activation EMRS625 [54]. The Copernicus EMS fire perimeters were derived from Planetscope (pre-fire 2022/06/01) and SPOT (post-fire 2022/08/20) very high-resolution satellite images.
The fire has burnt about 4.500 hectares, and almost 20.100 people have been evacuated from 9 towns and 2 campsites, as well as several injured people and road and railway service cuts.

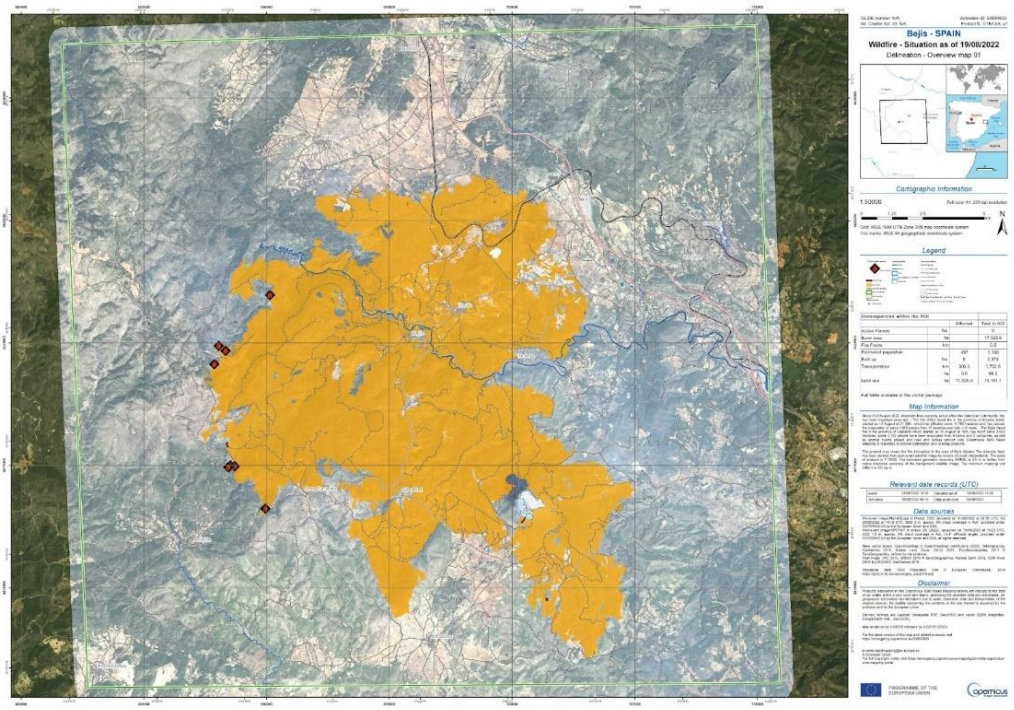


Figure 5.2 EMSR625 delineation map for Spain case study

The second test case was extracted from the severe forest fires that affected Chile, South America, in January-February 2023, identified by the Copernicus EMS-Mapping service activation EMSR647 [55].
The Copernicus EMS fire perimeters were derived from Sentinel 2 (pre-fire 2023/01/03) and SPOT (post-fire 2023/02/15) very high-resolution satellite images. In particular, the test was carried out over the area burned by the fire in the Nacimiento region south of the city of Concepción.

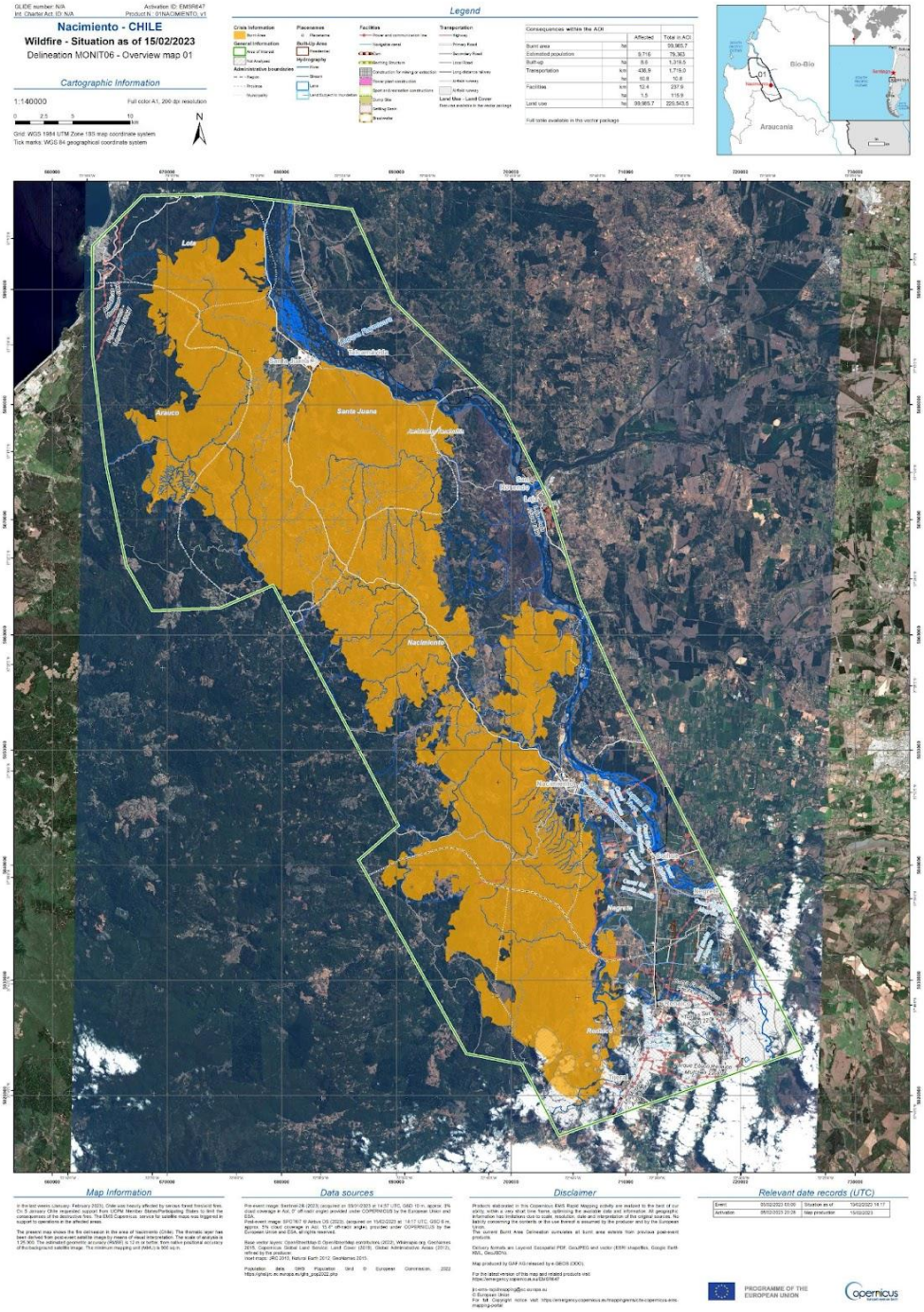


Figure 5.3 EMSR6747 delineation map for Chile case study

Some preliminary processing steps are necessary before running BAD plugin for the test case studies. First, the input data must be pre-processed:

- S-2 mission image data products processed after January 2022, have to be calibrated by applying an offset of 1000 introduced for each band; with respect to the previous pre-processing of S-2 data, this offset is necessary to generate

reflectance values consistent with the expected values of the parametrized Membership Functions within the CNR-IREA fuzzy algorithm. In order to do this a Python script was generated and implemented:

```
Raster_path= "*.tif"
Output_path_raster= "*.tif"

Raster_file = gdal.Open(Raster_path, gdal.GA_ReadOnly)
gt = Raster_file.GetGeoTransform()
proj= Raster_file.GetProjection()

Raster_matrices=Raster_file.ReadAsArray()
RasterMatrices=Raster_matrices.astype(float)
# applay an offset equal to -1000
for i in range (RasterMatrices.shape[0]):
   RasterMatrices[i]=RasterMatrices[i]-1000

WriteLayer(Output_path_raster,RasterMatrices,
RasterMatrices.shape[0],RasterMatrices.shape[2],RasterMatrices.shape[1],gt, proj)
```

Script 5.1 Offset calibration for each S-2 band

The script takes as input the raster's path, and reads the raster thanks to gdal library. After that, the bands are read as an array by the gdal's function and for each element of the array the offset (1000) quantity is subtracted. Finally, the output raster with the calibrated bands is saved with the *WriteLayer* function (described in Chapter 4).

This Python code is implemented to automate, speed up, and remedy the lack of a tool in QGIS capable of performing this procedure in "one shot": in QGIS it would be necessary to apply the correction to each band (with raster calculator) and then create a raster with the new bands;

- Second, a mask of non burnable regions is applied to the input raster images and to perform this step the following Python code is written.
  The non-burnable mask represents pixels that are not vegetated and hence cannot burn; this information is typically extracted from a land cover map of the study area. In these cases, the SCL layer (Table 1.1) provided with the S-2 images was used.

```
Raster_path= "*.tif"
SCL_path= "*.tif"
```

```
Output_path_mask= "*.tif"
Output_path_raster_masked= "*.tif"

Raster_file = gdal.Open(Raster_path, gdal.GA_ReadOnly)
SCL_file = gdal.Open(SCL_path, gdal.GA_ReadOnly)
gt = Raster_file.GetGeoTransform()
proj= Raster_file.GetProjection()

Raster_matrices=Raster_file.ReadAsArray()
SCL_matrix=SCL_file.ReadAsArray()

RasterMatrices=Raster_matrices.astype(float)
SCLMatrix=SCL_matrix.astype(float)

# MASK CREATION

MC1=np.where(SCLMatrix==1,np.nan,0)
MC2=np.where(SCLMatrix==2,1,0)
MC3=np.where(SCLMatrix==3,1,0)
MC4=np.where(SCLMatrix==4,1,0)
MC5=np.where(SCLMatrix==5,1,0)
MC6=np.where(SCLMatrix==6,np.nan,0)
MC7=np.where(SCLMatrix==7,1,0)
MC8=np.where(SCLMatrix==8,np.nan,0)
MC9=np.where(SCLMatrix==9,np.nan,0)
MC10=np.where(SCLMatrix==10,np.nan,0)
MC11=np.where(SCLMatrix==11,np.nan,0)

Mask= MC1+MC2+MC3+MC4+MC5+MC6+MC7+MC8+MC9+MC10+MC11
# applay the mask to each band of the raster

for i in range (RasterMatrices.shape[0]):
    RasterMatrices[i]=RasterMatrices[i]*Mask

# write the outputs

WriteLayer(Output_path_mask,Mask,1,Mask.shape[1],Mask.shape[0], gt, proj)

WriteLayer(Output_path_raster_masked,RasterMatrices,
RasterMatrices.shape[0],RasterMatrices.shape[2],
RasterMatrices.shape[1], gt, proj)
```

Script 5.2 Mask application for non burnable regions

In this script, main key features are:

- inputs raster files for both S-2 images and the SCL product;
- outputs generated: the mask layer and the masked raster;
- generation of the mask to be applied at each band:

Label	SCL classification	Mask value
0	No_data	nan
1	Satured_or_defective	1
2	Dark-area-pixels	1
3	Cloud_shadows	1
4	Vegetation	1
5	Not_vegetated	1
6	Water	nan
7	Unclassified	1
8	Cloud-medium-probability	nan
9	Cloud-high-probability	nan
10	Thin-cirrus	nan
11	Snow	nan

Table 5.1 Mask value derived from S-2 SCL layer

The mask, as already written in Chapter 3, is derived directly from the SCL layer within the Level 2A image data product and with the following criteria: all those pixels that represent an area that cannot burn (i.e. water), or pixels that should be discarded due to the presence of clouds, or an error during the collection of the data are masked out and assigned to not a number (nan). Table 5.1 summarizes the classes that were included in the mask layer.

5.1. Computer specification

The machine that is used to develop the work and run/test all process has the following characteristics:

- Processor: Intel(R) Core (TM) i7-7700HQ CPU @ 2.80GHz - 2.81 GHz;
- RAM: 16,0 GB;
- Operative system 64 bit.

5.2. Validation

In order to quantify accuracy of burned area maps derived with the BAD plugin, a validation of the obtained output is carried out. Validation is a critical and necessary task, as it provides a quantitative assessment of the accuracy of geo-information delivered by the product that is relevant for both scientists and end-users (Congalton and Green 1999 [56]). Accuracy is quantified with cross-tabulation between agreement and disagreement areas as derived from the comparison between a reference map and the output burned area map. Reference data should be independent, and in this thesis, Copernicus Emergency Management Service (EMS) fire perimeters were assumed as reference.
Cross-tabulation between reference and classified data generates the following table, called confusion matrix:

For both the three scenarios (Portugal, Spain, Chile) were estimated the following accuracy metrics derived from the confusion matrix:

- omission error (OE): it represents the fraction of values that belong to a class according to the reference but are predicted to be in a different class, it is a measure of false negatives;
- commission error (CE): it represents the fraction of values that are predicted to be in a class but do not belong to that class according to the reference, it is a measure of false positives;
- Dice coefficient (DC): it is a statistic coefficient used to evaluate the similarity of two samples and in this case, DC provides a single measure that combines the information of the class-specific metrics commission and omission errors [57];
- relative bias (relB): it is a value that quantifies the trueness of the method. It is also defined as the estimate of the systematic error and indicates whether a product overestimates (positive values) or underestimates (negative values) each class [57].

		REFERENCE	
		Burned	Unburned
BAD plugin estimation	Burned	True Positive (TP)	False Positive (FP)
	Unburned	False Negative (FN)	True Negative (TN)

Table 5.2 Confusion matrix representation

Accuracy Metric Name	Formula	Range
Omission Error	Oe=FN/(TP+FN)	[0,1]
Commission Error	Ce=FP/(TP+FP)	[0,1]
Dice Coefficient	DC=2TP/(2TP+FP+FN)	[0,1]
Relative Bias	relB = (FP-FN)/(TP+FN)	[-1,1]

Table 5.3 Metrics computed from the error/confusion matrix and range of variability

5.2.1. Accuracy assessment Portugal

For the first tests carried out over the Pedrógão Grande fire event, Portugal, the following subarea of the input S-2 image tile was used as input:

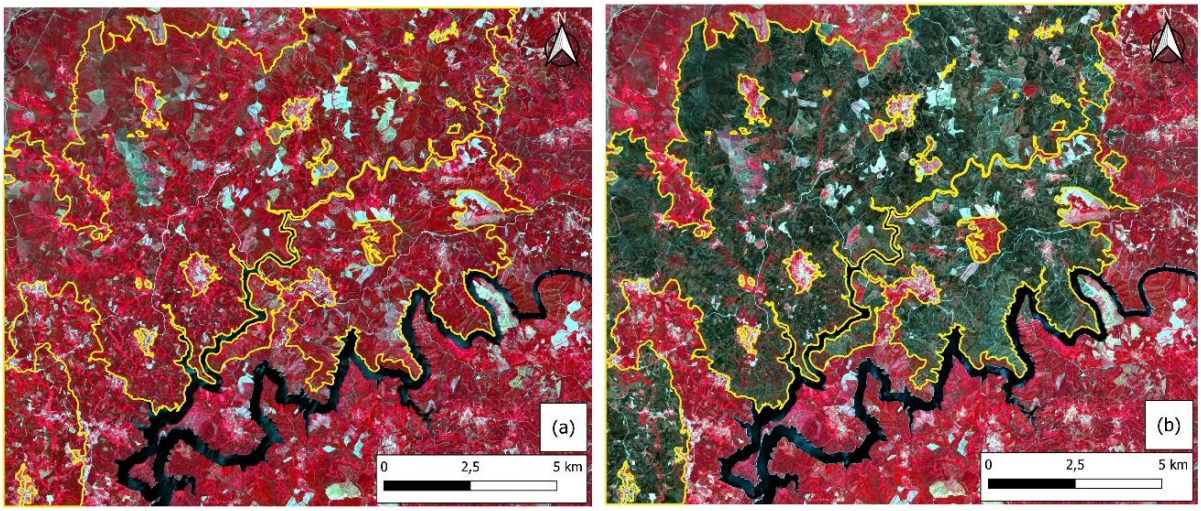


Figure 5.4 the S-2 images acquired before (2017-06-04) (a) and after (2017-07-04) (b) the wildfire that affected the Pedrógão Grande region, Portugal, in June 2017.
Images are shown in Color infrared to highlight vegetation (RGB false color combination is NIR-Red-Green). The raster dimension is 1705x1445 pixels and pixel size is 10m.
EMSR reference perimeter (yellow line)

The BAD plugin provided the outputs shown in the figures below: burn severity map (Figure 5.5), RG burned area map (Figure 5.6) and Burn Severity Map for the identified burned pixels (Figure 5.7). The burn severity map shows the classification derived by applying the dNBR algorithm (section 2.3) as degrees of burn severity (yellow to orange) and vegetation regrowth (green); areas masked out as non burnable regions are shown in gray.

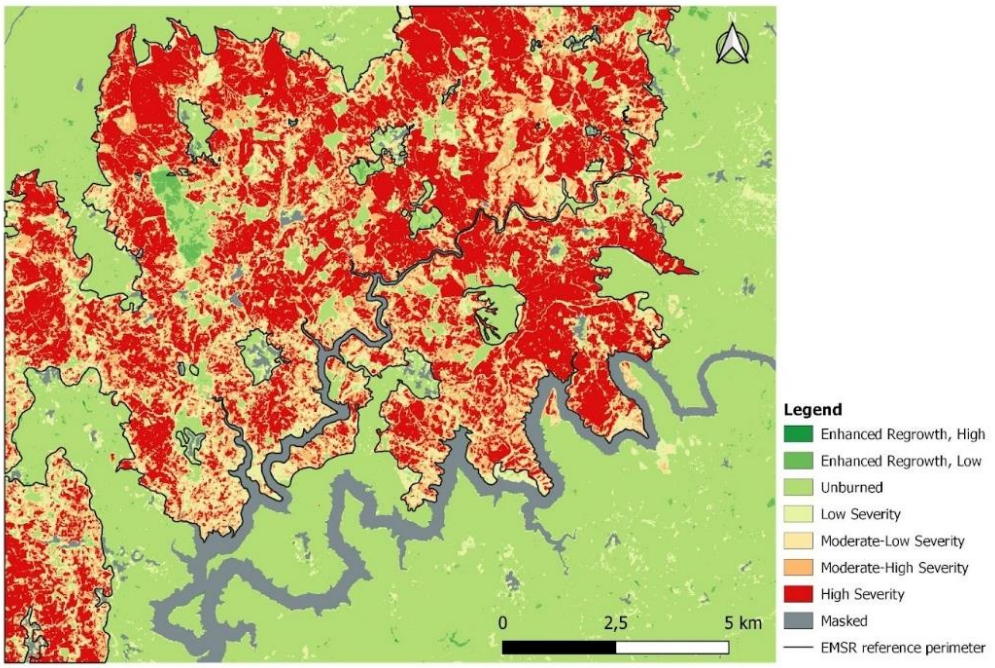


Figure 5.5 Severity Map obtained applying dNBR algorithm for Portugal case study

The RG output shows the burned area map derived from the BAD classification; this map contains three classes: burned (red), unburned (green) and masked (gray).

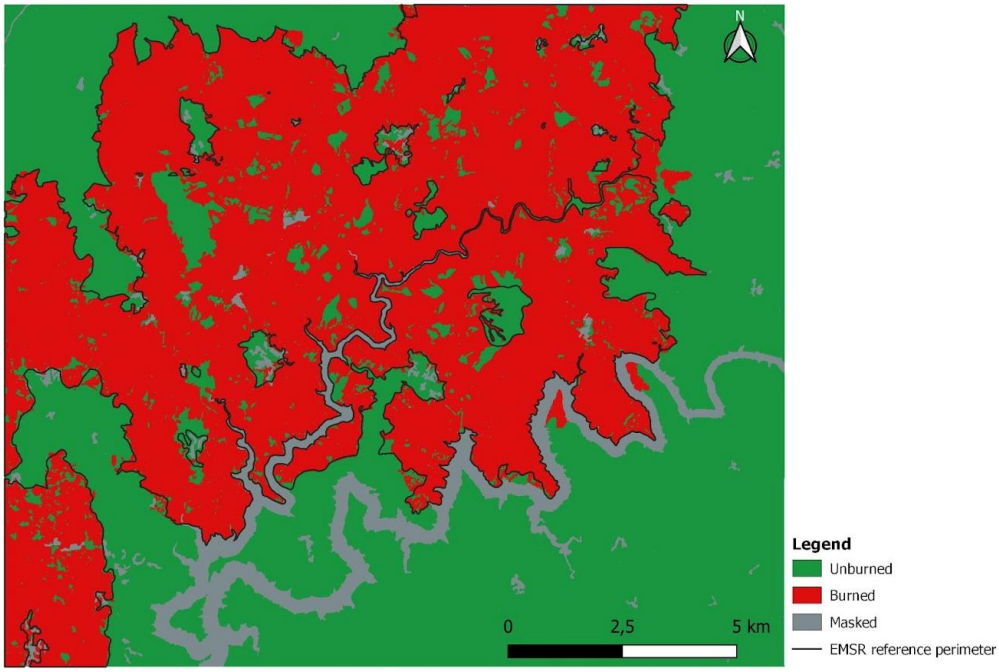


Figure 5.6 Burned Area Map obtained applying RG algorithm for Portugal case study

The final output map from BAD is the burn severity for the identified burned pixels. This map is obtained as combination of the previous two outputs (Severity and RG map), therefore for each pixel it is possible to retrieve the following information:

- if a pixel represents an area on the ground that is burned or not;
- if the pixel is flagged as burned, the severity level (level of damage) associated to that ground area.

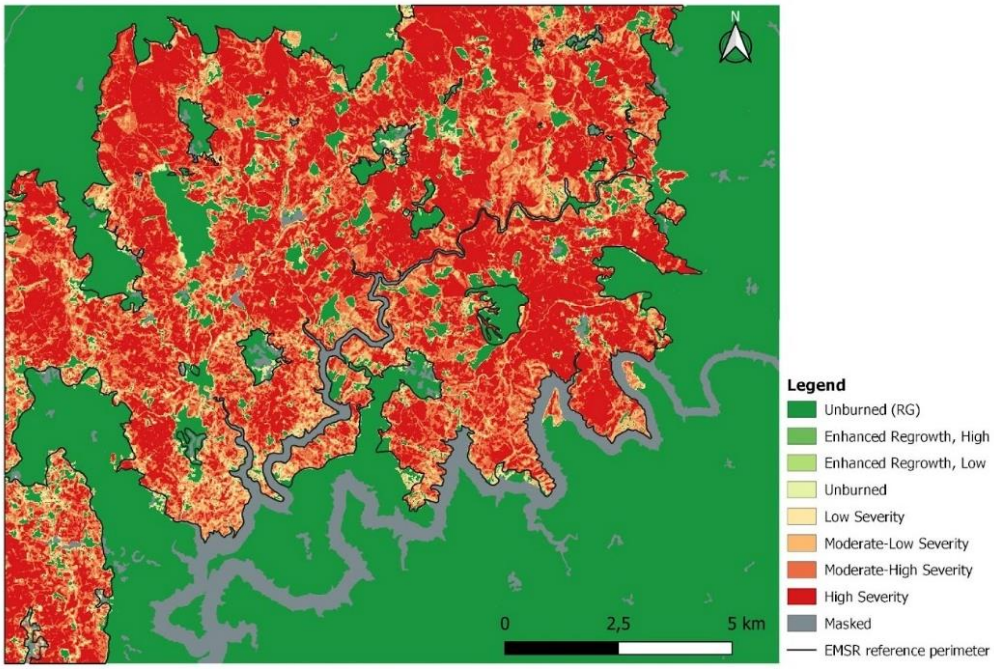
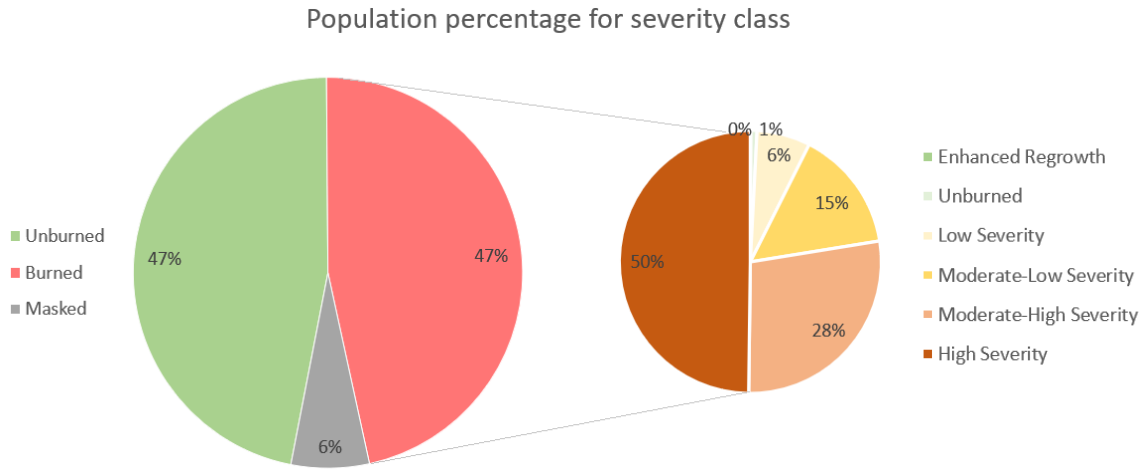


Figure 5.7 Burn Severity Map for the identified burned pixels for Portugal case study

This map, that shows the Burned class within the study area (47% of the total number of pixels), shows the pixels classified in the different classes provided by the dNBR algorithm. In fact, counting the number of burned pixels for each severity class:

- more than 99% of the "Burned" population has level of severity from low (class 4, yellow) to high (class 7, red);
- the population with a moderate-high/high level of severity (class 6 and 7, respectively) is about 78%;
- less than 1% of the population is classified as "Unburned" (class 3);
- less than 0.1% of the population has classified as enhanced regrowth (class 1 and 2).



Charts 5.1 On the left the pie chart represents the population (in percentage) in each class according to the RG output; on the right the chart represents the severity class only for those pixels that are classified as "Burned" by the fuzzy BA algorithm implemented in the BAD plugin for Portugal case study

The presence of pixels classified as "Burned" according to RG but with a severity level between 1 (Enhanced-Regrowth, High) and 3 (Unburned) may seem contradictory and/or counterintuitive. However, this result, which represents a clear minority of the population (about 1%), is due to the fact that the two classifications are independent hence they could provide a different classification label. The fact that only a negligible proportion (~1%) of the "Burned" pixels identified by the fuzzy BA algorithm falls in the unburned class of the dNBR algorithm shows that output results are accurate and reliable.
The independent validation carried out by comparison with CEMS fire perimeters further confirms that commission errors are very low (Table 5.06).

The results of the pixel by pixel comparison between BAD out map and the reference can be also presented as an agreement map, that depicts the spatial distribution of agreement (red and green) and disagreement areas. Disagreement is shown as omission (yellow) and commission (blue) errors (Figure 5.8). This map describes the relation between what is produced by BAD plugin and the reference data (CEMS). During this pixel by pixel comparison the cases that can occur are:

- pixels assigned to the same class by the algorithm and by the reference: Not Burned (green) (True Negative), Burned (red) (True Positive) ;
- pixels classified into different classes. In this scenario a distinction must be done:
    - a False Negative is produced: plugin classifies the pixel as Not Burned, instead EMS as Burned, that is an omission error (yellow);

- a False Positive is produced: plugin classifies the pixel as Burned, instead reference as Not Burned, that is a commission error (blue).

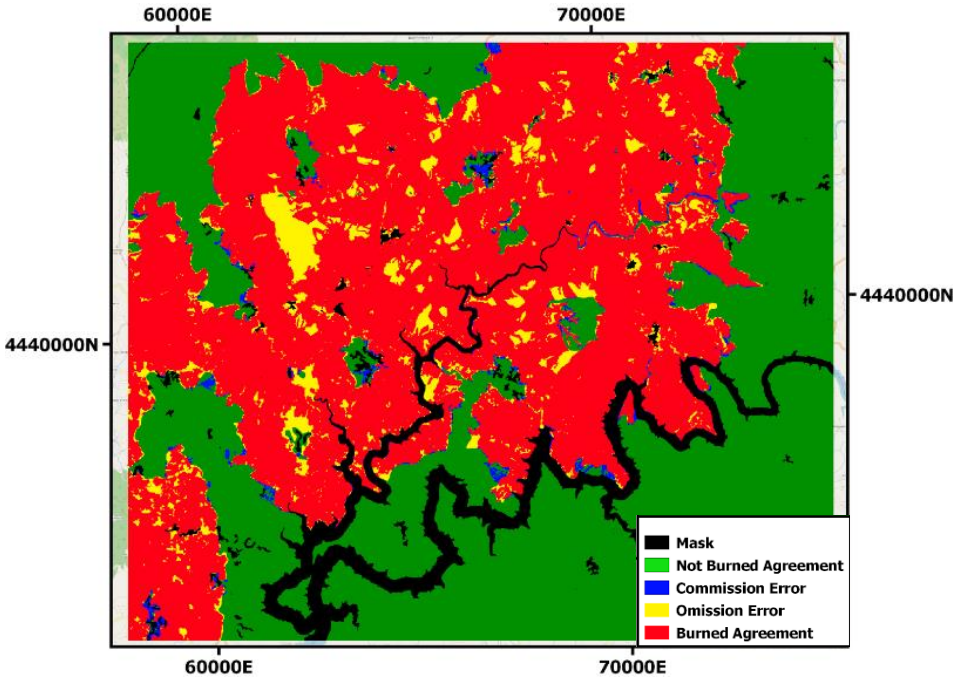


Figure 5.8 Agreement map for Portugal case study

Here are reported the values obtained from the EMS site and the values derived from the plugin, finally the accuracy metrics are computed:

- data from EMS and form the BAD plugin:

	Pixels area [m²]
EMS BA	124393300
EMS BA masked	123399100
RG BA	115038300
Mask	15964800
Region Area	246372500

Table 5.4 Data from EMS and form BAD plugin for Portugal case study

- Confusion matrix:

		REFERENCE	
		Burned	Unburned
BAD plugin estimation	Burned	112769100	2269200
	Unburned	10630000	104739400

Table 5.5 Confusion matrix for Portugal case study

- Accuracy parameters: applying the formula reported in Table 5.3 is possible to obtain these metrics:

	%
Omission Error	9,24
Commission Error	1,84
Dice Coefficient	94,59
Relative Bias	-6,78

Table 5.6 Accuracy metrics for Portugal case study

The metrics representing the two errors are low: Oe=9.24% and Ce=1.84%.

Therefore, in this case study the error of omission is greater than the commission one as confirmed by the negative value of RelB metric. This means that: the extension of burned areas, erroneously classified by the plugin as unburned (in yellow Figure 5.8 agreement map), are greater than the amount of area classified as burned by BAD but not burned in the reference (in blue Figure 5.08 agreement map). Omission errors in the agreement map appear to be mainly due to unburned islands inside the major burned perimeter.
Instead, observing the Dice coefficient (equal to 94,59%, close to 1) and the relative bias (-6.78%, close to 0) that the output generated by BAD plugin is similar to the reality of the event described by the reference produced by EMS.
Notice that CEMS fire perimeters are derived from very high resolution satellite images (in the case of the Portugal fire event from SPOT 7) hence commission errors in the BAD output are expected due to the lower spatial resolution of the input S-2 images (10-20 m).

5.2.2. Accuracy assessment Spain

The procedure applied for computing the accuracy assessment over the study area in Spain is similar to what was done for Portugal, therefore in this paragraph are reported only the principal step, the main difference and the commented results.

BAD plugin input are pre-fire and post-fire images shown in Figure 5.9.

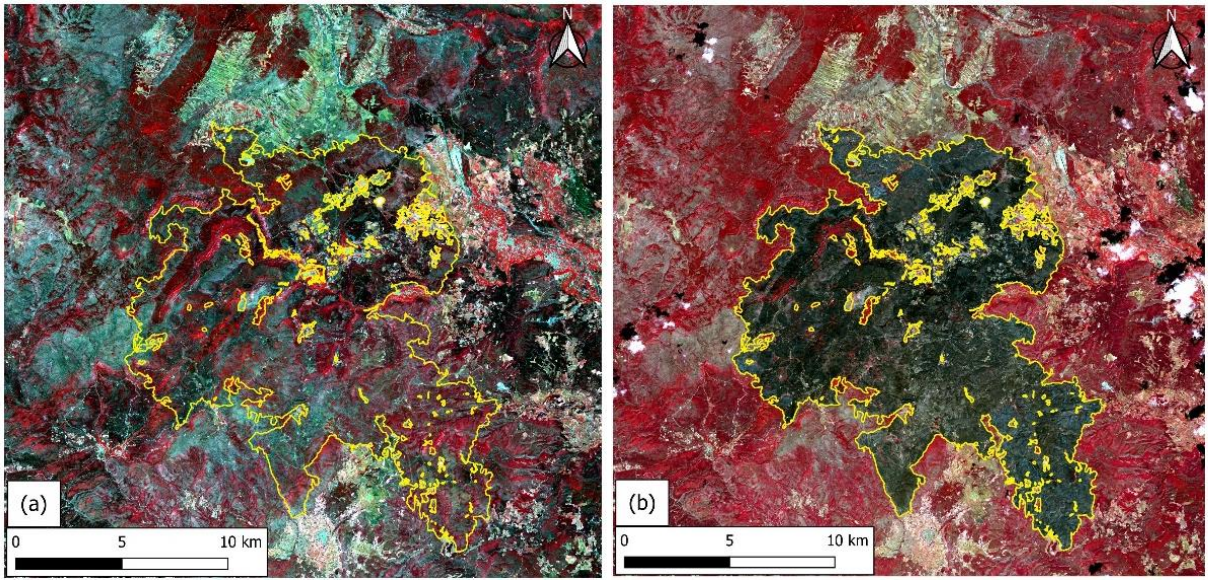


Figure 5.9 the S2 images acquired before (2022-07-04) (a) and after (2022-08-23) (b) the wildfire that affected the Bejís region, Spain, in August 2022. Images are shown in Color infrared to highlight vegetation (RGB false color combination is NIR-Red-Green). The raster dimension is 2929x2706 pixels and pixel size is 10m. EMSR reference perimeter (yellow line)

Applying BAD on the subarea of the event, the following output products are derived: burn severity map (Figure 5.10), RG burned area map (Figure 5.11) and severity map (Figure 5.12) for the identified burned pixels.

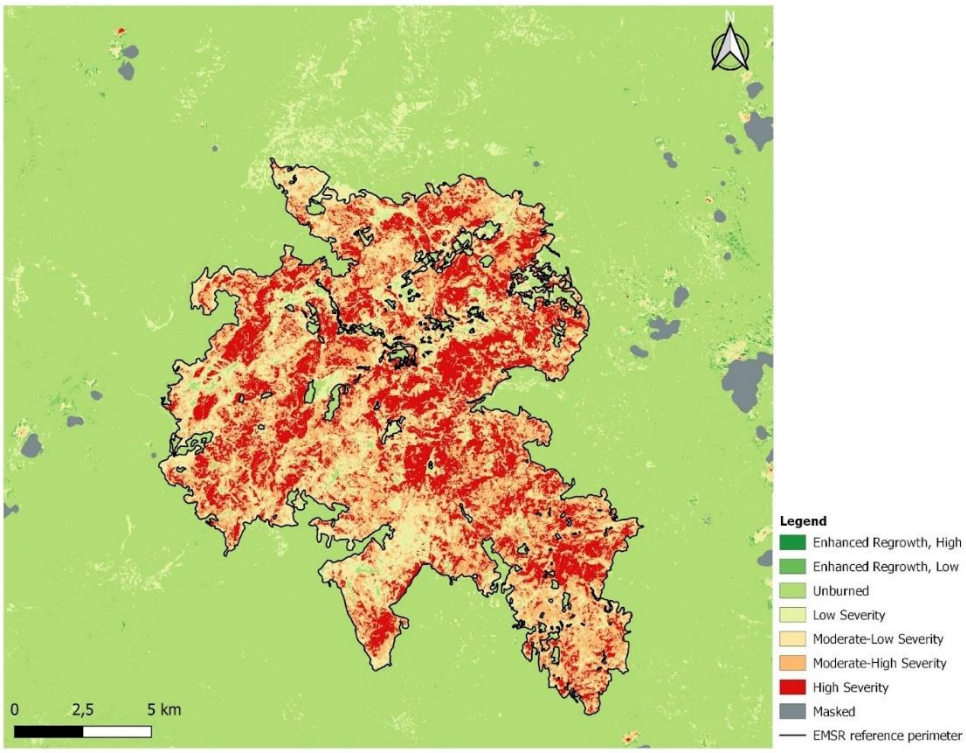


Figure 5.10 Severity Map obtained applying dNBR algorithm for Spain case study

Figure 5.11 Burned Area Map obtained applying RG algorithm for Spain case study

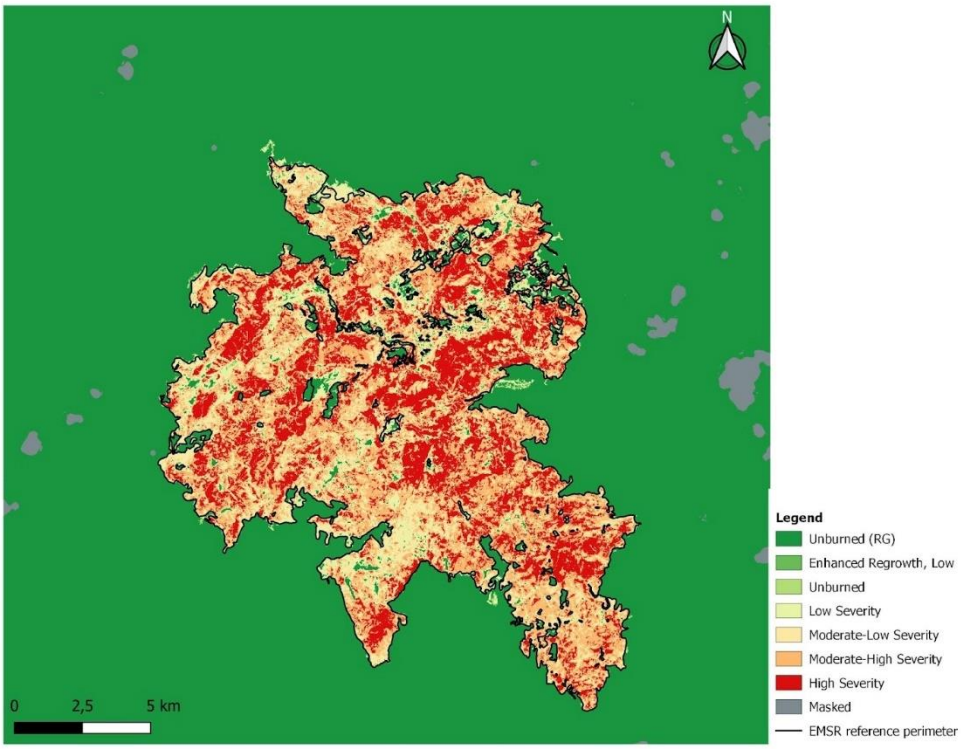
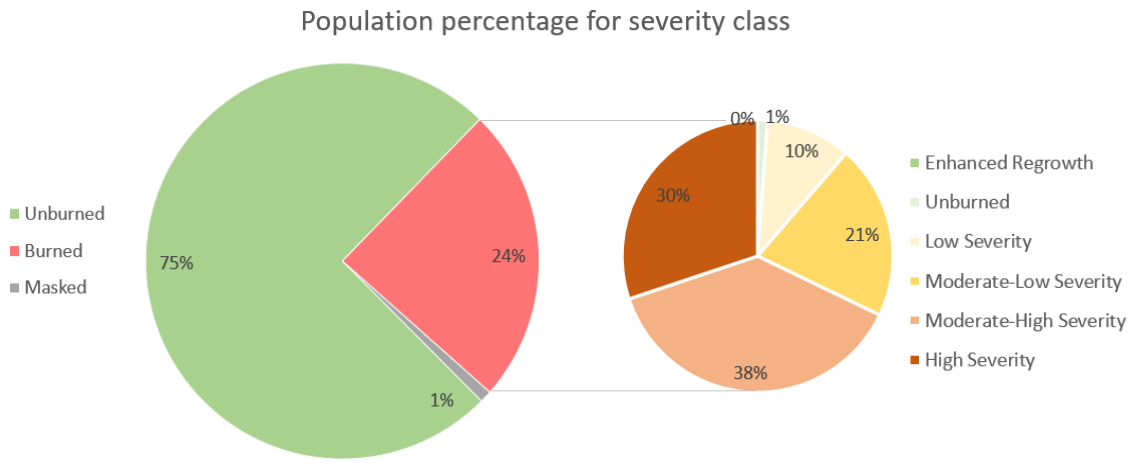


Figure 5.12 Burn Severity Map for the identified burned pixels for Spain case study

According to the last map (Figure 5.12), within the Burned class (24% of the total number of pixels) it could be expected to find pixels with a high level of severity.
In fact, counting the number of burned pixels for each severity class:

- more than 99% of the population has level of severity from low (class 4, yellow) to high (class 7, red);
- the population with a moderate-high/high level of severity (respectively class 6 and 7) is about 68%;
- less than 1% of the population has classified as "Unburned" (class 3, light green);
- less than 0.01% 0% of the population has classified as enhanced regrowth (class 1 and 2).



Charts 5.2 On the left the pie chart represents the population (in percentage) in each class according to the RG output; on the right the chart represents the severity class only for those pixels that are classified as "Burned" by the fuzzy BA algorithm implemented in the BAD plugin for Spain case study

According to reference data from Copernicus EMS, it is possible to make the comparison with the BAD output:

- Data:

	Pixels area [m²]
EMS BA	180583500
EMS BA masked	180572600
RG BA	184393700
Mask	7384200
Region Area	761187000

Table 5.7 Data from EMS and form BAD plugin for Spain case study

- Confusion matrix:

		REFERENCE	
		Burned	Unburned
BAD plugin estimation	Burned	176231000	8162700
	Unburned	4341600	565067500

Table 5.8 Confusion matrix for Spain case study

- Accuracy parameters:

	%
Omission Error	2,40
Commission Error	4,43
Dice Coefficient	96,57
Relative Bias	2,12

Table 5.9 Accuracy metrics for Spain case study

The metrics representing the two errors are low: Oe=2.40% and Ce=4.43%.

The error of omission is lower than the commission one. This means that: the extension of burned areas, misclassified by the plugin as "Unburned" (in yellow Figure 5.13), are smaller than the amount of area classified as "Burned" by BAD but not burned in the reference (in blue Figure 5.13). This is confirmed by the relative bias metric (RelB=2.12%) that, even if negligible, is a positive value meaning that commission is greater than omission. One source of disagreement between the EMS fire perimeter and the BAD output could be the date of the post-fire images used for BAD and reference. Indeed, if small fires are actively burning, even a few days difference (in this case three days), might lead to a greater amount of area burned.

The Dice coefficient (DC=96.57%, close to 100%) confirms that the output generated by the BAD plugin is similar to the reference as described by the Copernicus EMS perimeter hence results are more than satisfactory.

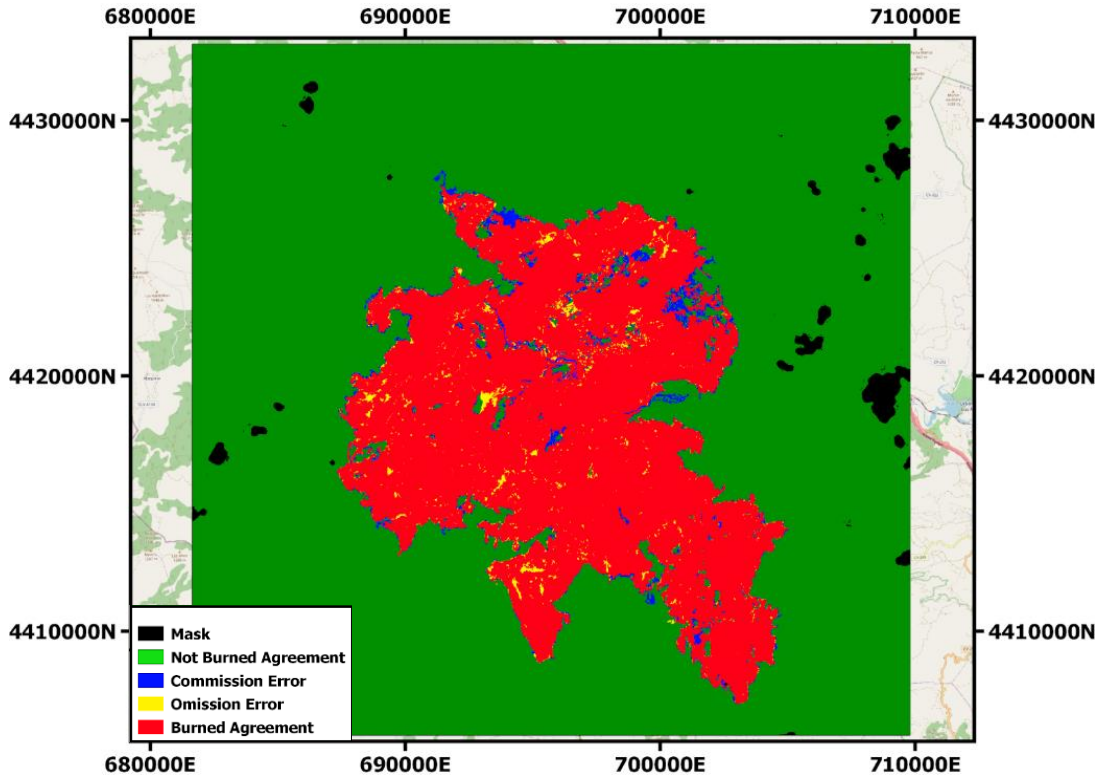


Figure 5.13 Agreement map

Another aspect that can be observed after a first look at the images, in this case study, is that there are several pixels outside the main burned perimeter, which were identified as "low severity" by the dNBR algorithm (yellow regions in Figure 5.10). Therefore, it can be useful to verify if the approach used during this thesis work (the Region Growing approach offered by BAD) generates a better output than the dNBR algorithm by comparing both output maps to the reference fire perimeters.
The confusion matrices generated the error accuracy metrics shown in table 5.10. Except for the commission error, all the metrics are better when BAD is used; overall DC=96,57% for BAD thus confirming that more accurate results are achieved with the fuzzy burned area mapping algorithm implemented in the plugin.

	dNBR [%]	BAD [%]
Omission Error	17,08	2,40
Commission Error	1,76	4,43
Dice Coefficient	91,12	96,57
Relative Bias	15,05	2,12

Table 5.10 Accuracy metrics for Spain case study: dNBR vs BAD result

5.2.3. Accuracy assessment Chile

The final test case study of forest fires in Southern America, Chile, used the following input S-2 images before and after the fire.

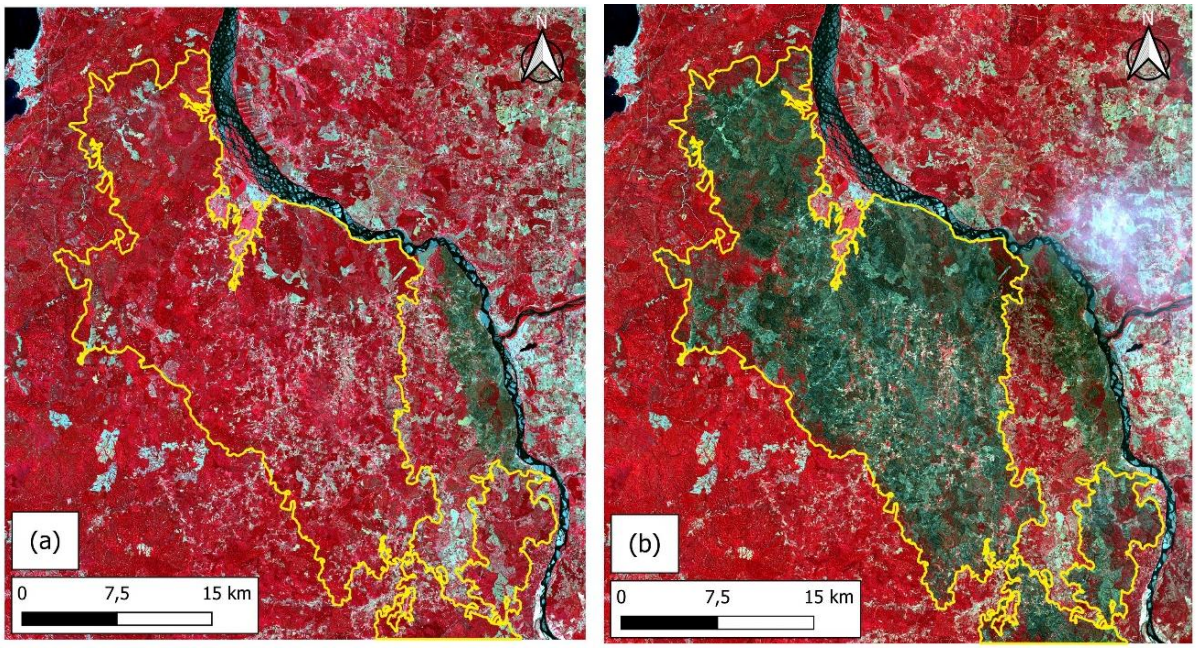


Figure 5.14 The S-2 images acquired before (2023-01-28) (a) and after (2023-02-17) (b) the wildfire that affected the Nacimiento region, Chile, in February 2023. Images are shown in Colour infrared to highlight vegetation (RGB false colour combination is NIR-Red-Green). The raster dimension is 4587x4986 pixels and pixel size is 10m. EMSR reference perimeter (yellow line)

Applying BAD plugin on subarea of the event following output are produced:

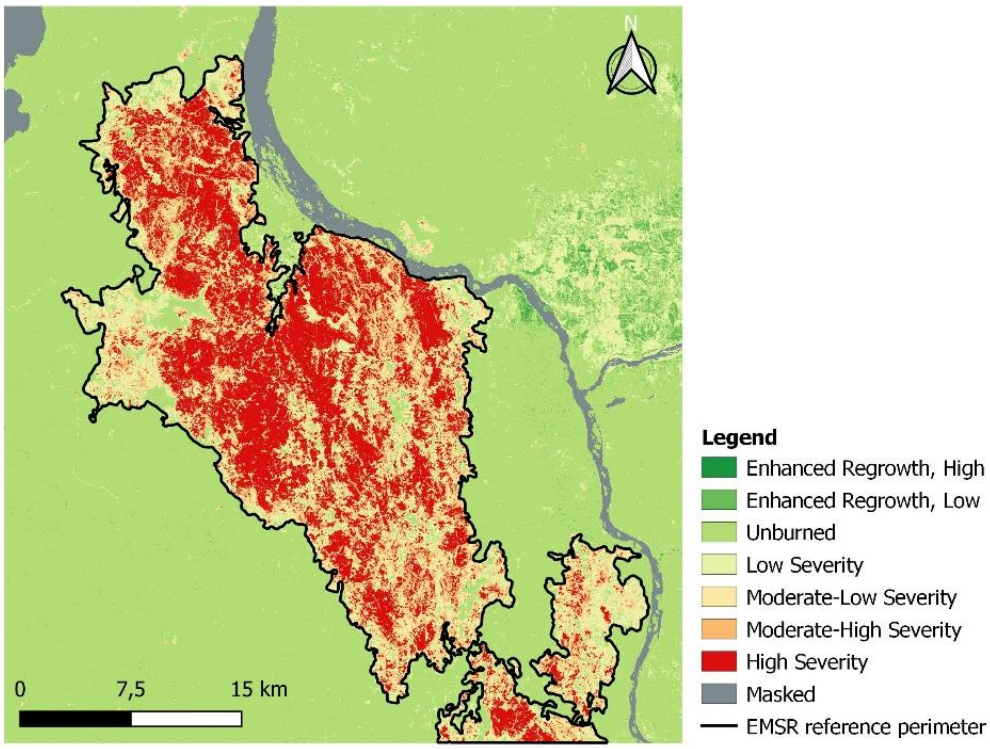


Figure 5.15 Severity Map obtained applying dNBR algorithm for Chile case study

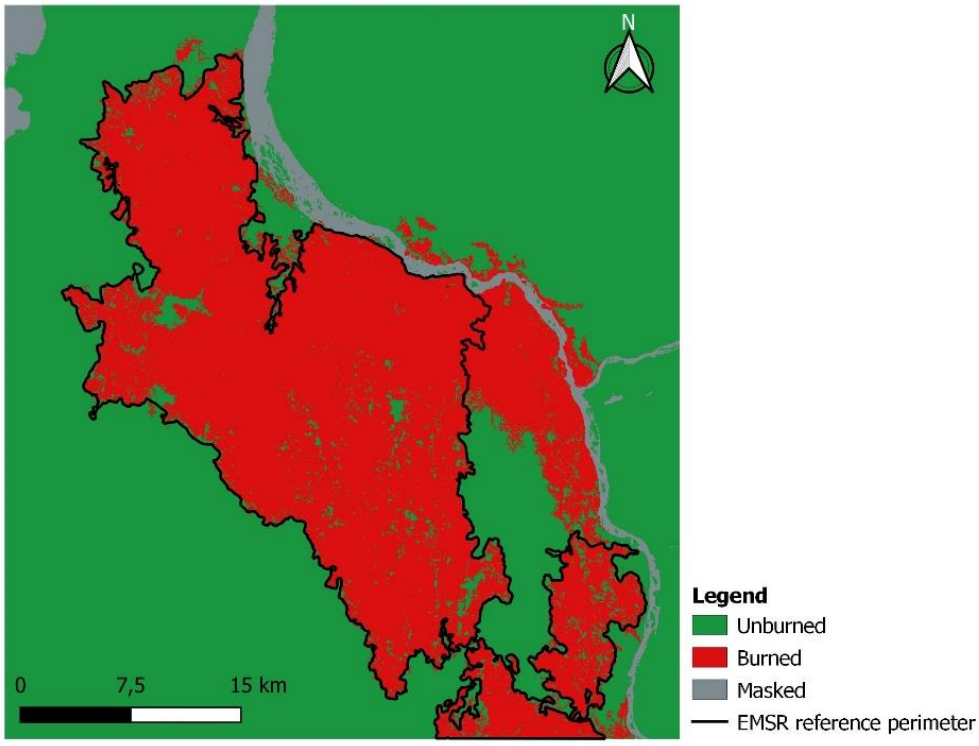


Figure 5.16 Burned Area Map obtained applying RG algorithm for Chile case study

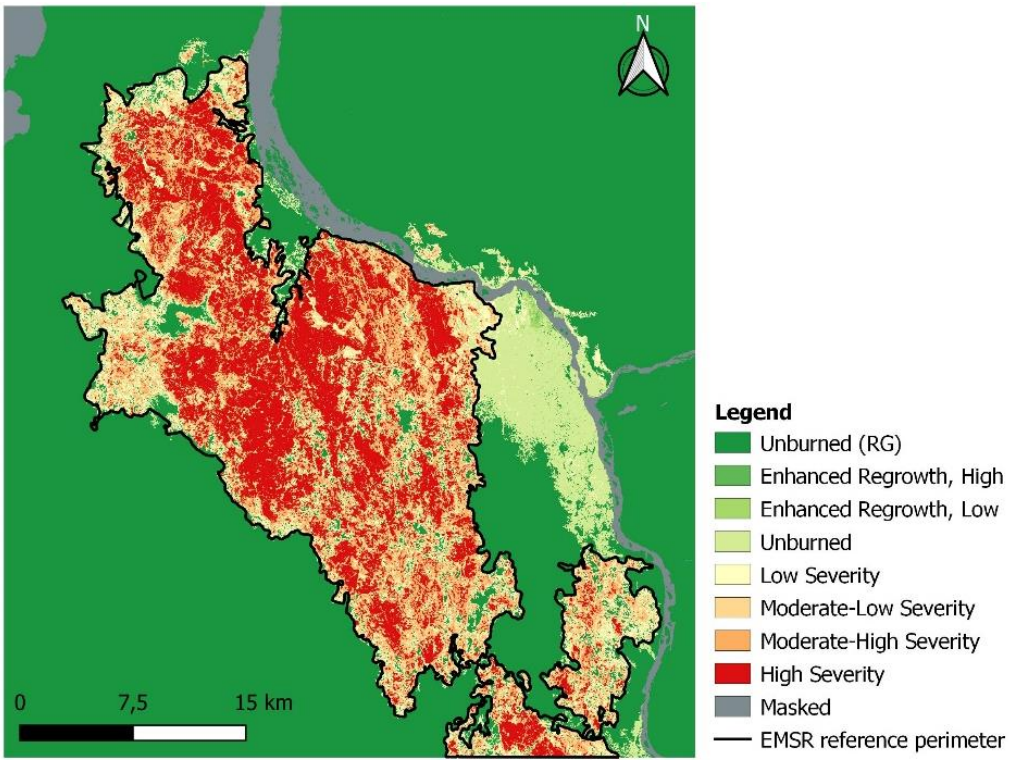
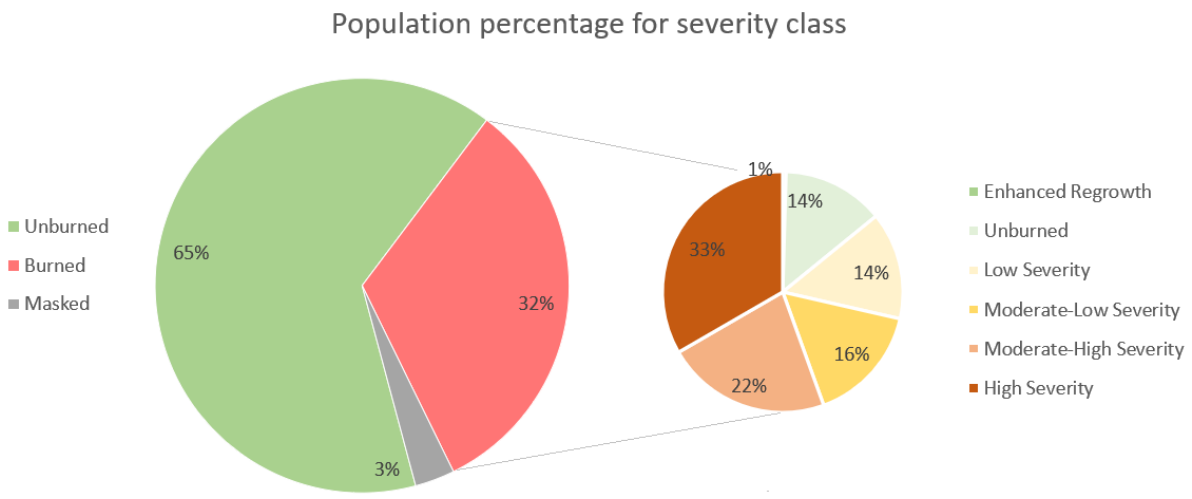


Figure 5.17 Burn Severity Map for the identified burned pixels for Chile case study

According to the last map (Figure 5.17), within the Burned class (32% of the total number of pixels) it could be expected to find pixels with a high level of severity.
In fact, counting the number of burned pixels for each severity class:

- more than 84% of the population has level of severity from low (class 4, yellow) to high (class 7, red);
- the population with a moderate-high/high level of severity (respectively class 6 and 7) is about 55%;
- about 14% of the population has classified as Unburned (class 3);
- less than 1% of the population has classified as enhanced regrowth (class 1 and 2).



Charts 5.3 On the left the pie chart represents the population (in percentage) in each class according to the RG output; on the right the chart represents the severity class only for those pixels that are classified as "Burned" by the fuzzy BA algorithm implemented in the BAD plugin for Chile case study

Looking for the data present inside the official page of the event (reference data) it is possible to make the comparison with the BAD output:

- Data:

	Pixels area [m²]
EMS BA	685255600
EMS BA masked	685255400
RG BA	740369800
Mask	54797900
Region Area	1466400900

Table 5.11 Data from EMS and form BAD plugin for Chile case study

- Confusion matrix:

		REFERENCE	
		Burned	Unburned
BAD plugin estimation	Burned	633042100	107327700
	Unburned	52213300	619019900

Table 5.12 Confusion matrix for Chile case study

- Accuracy metrics:

	%
Omission Error	7,05
Commission Error	15,66
Dice Coefficient	88,81
Relative Bias	8,04

Table 5.13 Accuracy metrics for Chile case study

Finally, the agreement map can be plot:

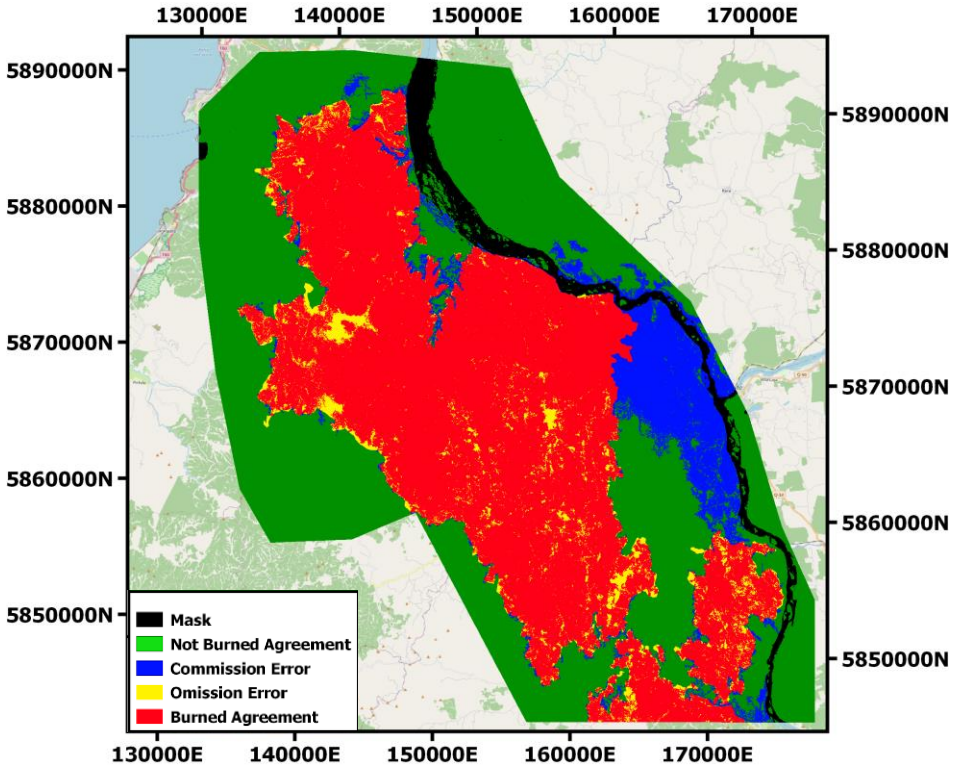


Figure 5.18 Agreement map for Chile case study

For this case study, the commission is significantly higher: Ce=15.66%, if compered to values obtained from the validation of Portugal (1.84%) and Spain (4.43.%).

The main region where the Ce was made are highlighted in Figure 5.18 (blue area). These areas are not burned according to the reference perimeter (Figure 5.14, yellow line) proposed by EMS, because it was already burnt on pre-fire dates.
This is indeed confirmed by the false colour composites of the S-2 images (Figure 5.14 (a)), where the area is apparently burned in the first S-2 image; therefore, this area should not be burned between the two dates hence should not be detected.
In fact, in Figure 5.14 (b), the anomaly is still present, in addition to the new one (area inside the EMS reference perimeter, yellow line).
This aspect is also highlighted by the Severity Map (Figure 5.15), where the area was mainly classified as "Unburned" due to the fact that any changes were found by the severity mapping algorithm (paragraph 2.3)
However, the BAD plugin classified that area as "Burned". The reason is due to the features selected, in particular the post ones, and the OWA operator (OWA_OR)

used in order to integrate the information and generate the grow layer.
It is likely that the post-fire features have a greater MD value than delta features thus being the best candidates in the OWA_OR, which tends to select features with the greatest MD value. Being the burned spectral signal in the post-fire image still clearly visible, this area is probably added in the growing phase together with the new burns.
These choices made the pixels, in that area, good candidates to become new seeds, and since there were seeds that fell within the area, the RG has classified those pixels as "Burned"; while according to the reference they are not "Burned".
This aspect influenced the Ce computation.

5.3. Performance

The main variable that influences the computational time is the size of the images (i.e. raster format number of rows by number of columns): the greater the image size, the longer the time to read/write (I/O) the data, to compute operations between the arrays, and to apply Region Growing.

This last step of the algorithm is the most expensive in terms of computing time. The RG algorithm, as well explained in Chapter 3, is an iterative process, and, as reported in paragraph 4.4.5.2 (how RG is implemented), at each iteration two layers (Seed and Grow) are read pixel by pixel and conditions are tested for each pixel and finally a matrix is written (Burned result). This aspect is shown in Charts 5.4, but much more evident in Charts 5.5 where the effort of each phase is reported in terms of time [seconds] and as a percentage of total computation time.

Thus, looking at Charts 5.4 and Charts 5.5, it can be seen that as the size of the images (dimension of the arrays) increases, more time will be required to execute the process.

To show the behavior of the computational time, seven images, having a pixel resolution of 10x10 m, with different sizes are considered. These images are obtained by extracting a subset S-2 images over a wildfire event that occurred in Portugal (pre-fire: 4th June 2017, post-fire: 4th July 2017); the test images used for quantifying running time are a square array of n x n rows and columns: seven different image size (n) are tested varying from 2x2 km to 14x14 km, with a step of 2 km (200 pixels) as summarized in Table 5.14.

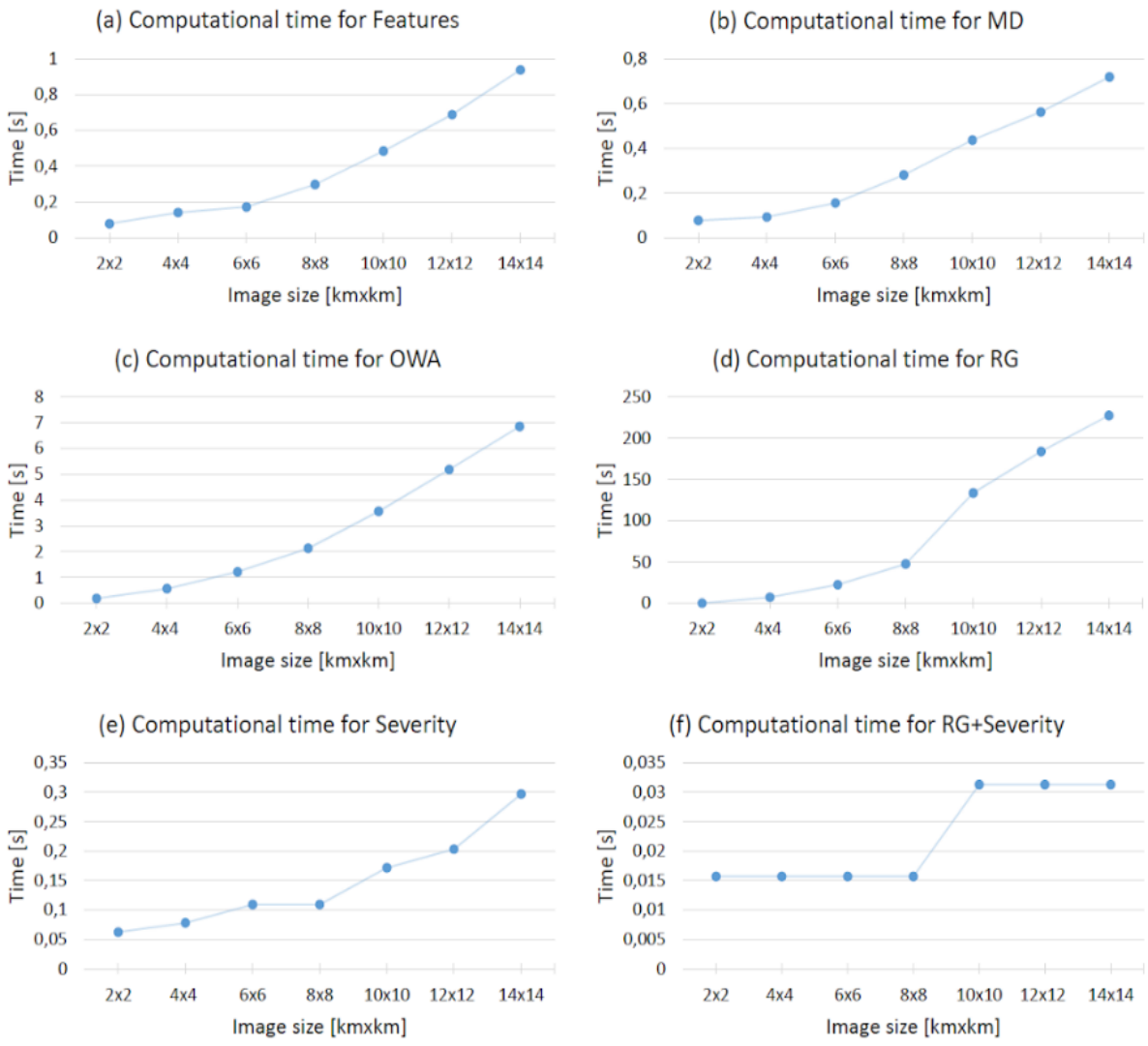
Computational time is measured for the following steps/operations within the burned area mapping algorithm implemented in BAD plugin:

- Compute Feature;
- Compute Membership Degree (MD);
- Compute OWA (OWA_AND and OWA_OR);
- Compute Region Growing;
- Compute the Severity;
- Compute Severity combined with Region Growing result.

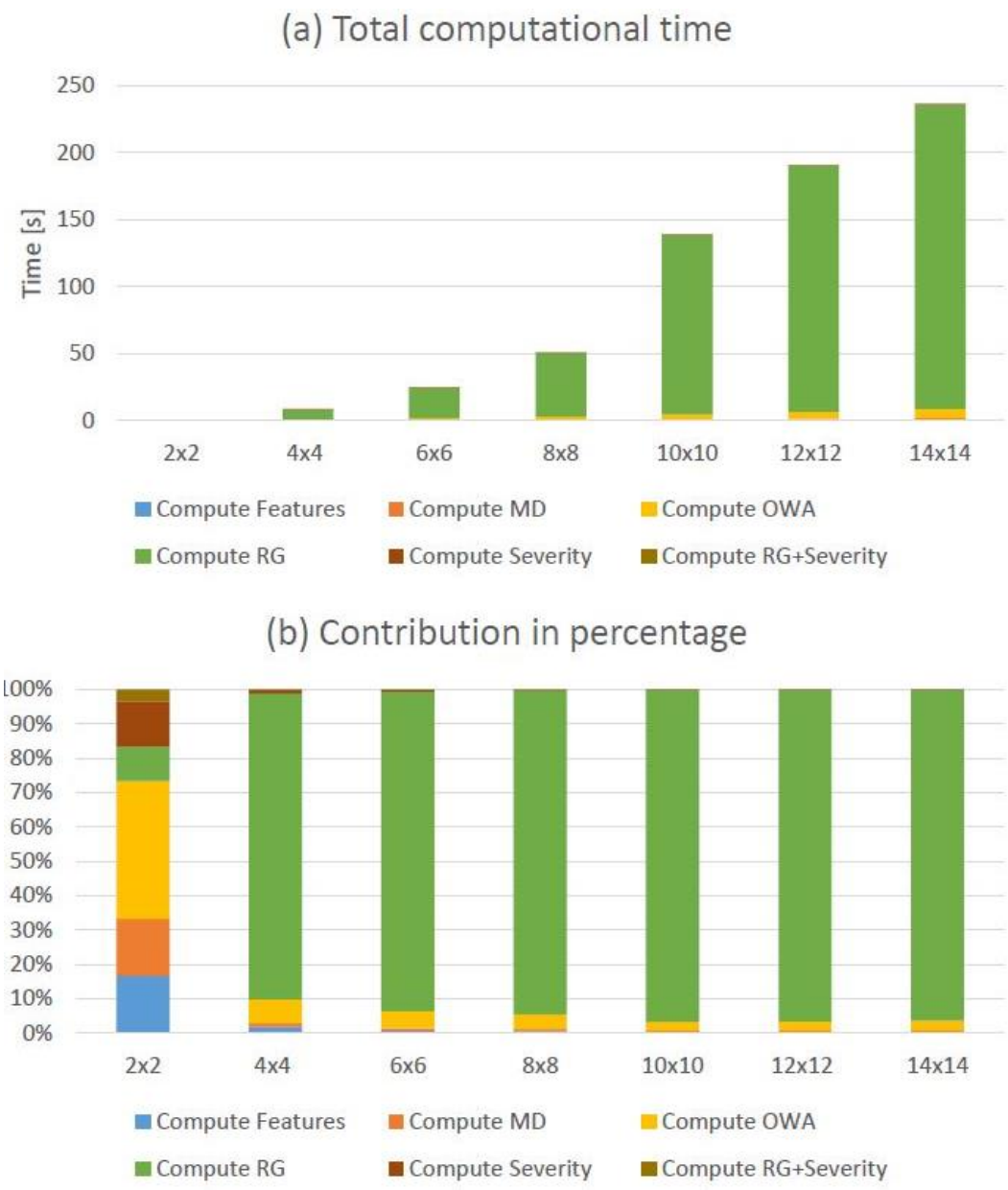
The result [seconds] are reported in table Table 5.14:

	Image size NxN [Km x Km]						
	N=2	N=4	N=6	N=8	N=10	N=12	N=14
Compute Features	0,08	0,14	0,17	0,30	0,48	0,69	0,94
Compute MD	0,08	0,09	0,16	0,28	0,44	0,56	0,72
Compute OWA	0,19	0,56	1,22	2,13	3,56	5,19	6,86
Compute RG	0,05	7,22	22,61	47,55	133,83	183,90	227,58
Compute Severity	0,06	0,08	0,11	0,11	0,17	0,20	0,30
Compute Severity+RG	0,02	0,02	0,02	0,02	0,03	0,03	0,03
TOT [s]	0,48	8,11	24,29	50,39	138,51	190,57	236,43

Table 5.14 computational time for each BAD's operation for different size subareas; images size goes from 2x2km to 14x14 km with a step of 2x2km for Portugal case study



Charts 5.4 computational time [s] for each BAD's operation for different size subareas; images size go from 2x2km to 14x14 km with a step of 2x2km for Portugal case study

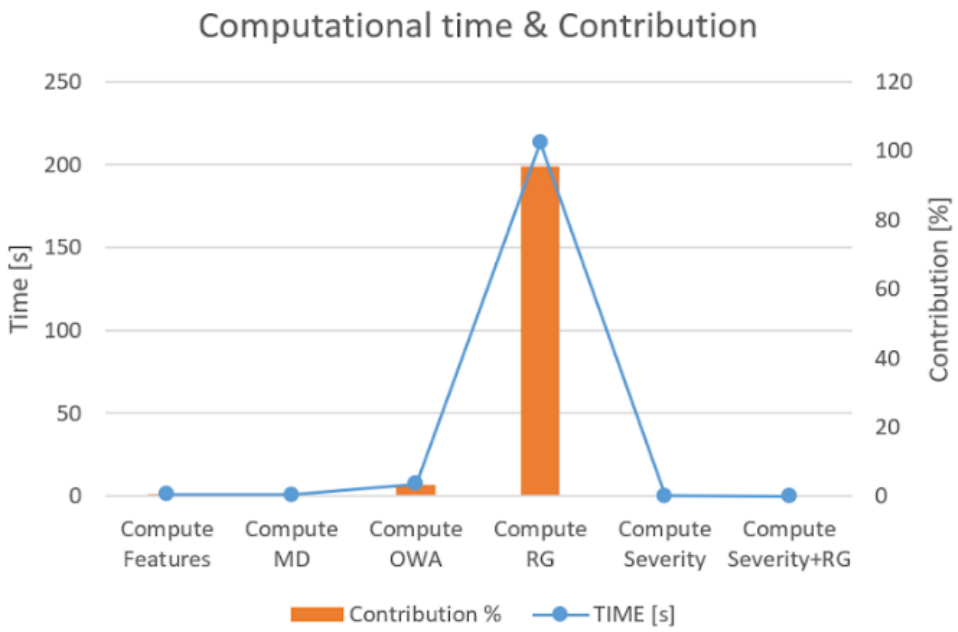


Charts 5.5 Total amount of BAD computational time for different size subareas distinguish between the effort of each operation in: time[s] (a); percentage [%] (b) for Portugal case study

To confirm trends shown in Charts 5.5, considering the entire test images for the fire event in Portugal Figure 5.04; 17,05x14,45 km), the total amount of time required by the BAD plugin is 3 minutes and 43 seconds, and 95,46% of this time is required for the Region Growing step.

	Time [s]	Contribution [%]
Compute Features	1,20	0,54
Compute MD	1,02	0,45
Compute OWA	7,56	3,38
Compute RG	213,28	95,46
Compute Severity	0,31	0,14
Compute Severity+RG	0,05	0,03
TOT	223,42	100

Table 5.15 Computational time for each BAD's operation for the entire study area in Portugal (17,05x14,45 km)



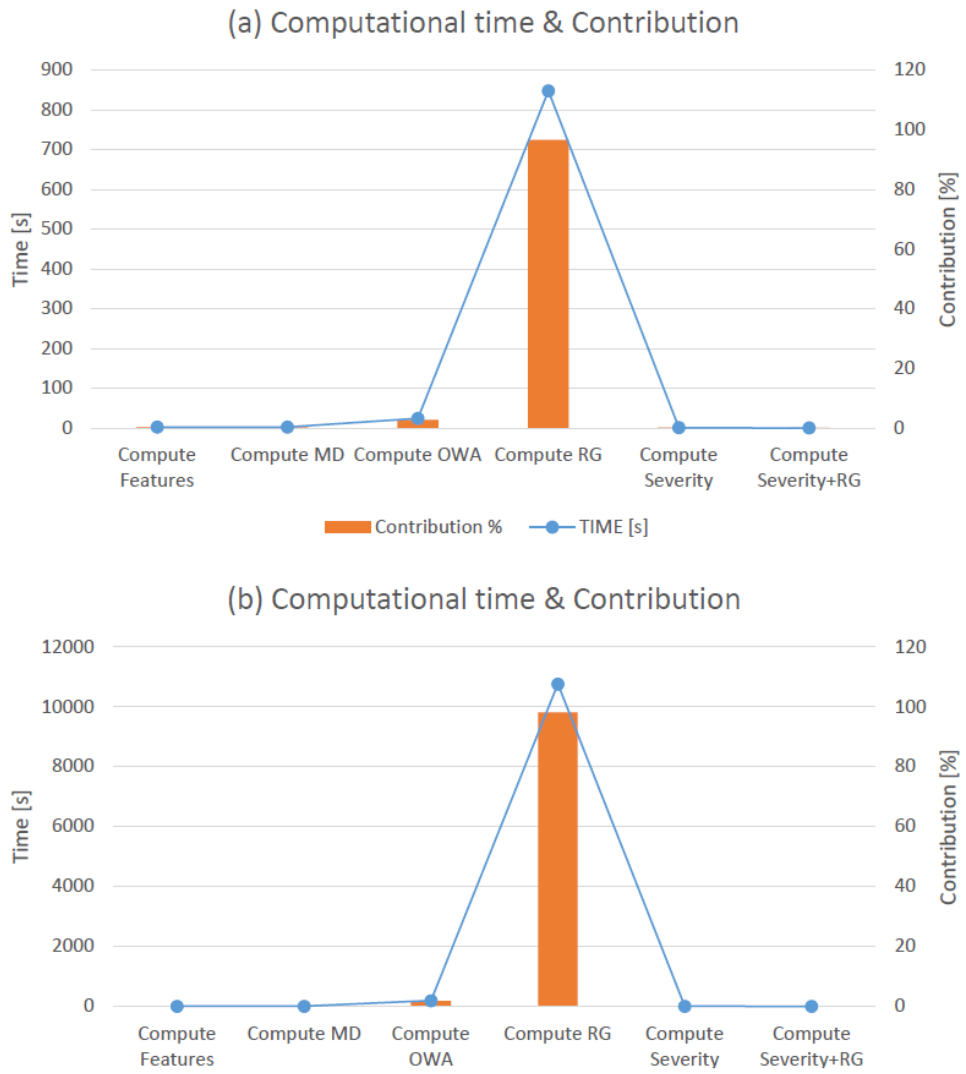
Charts 5.6 Computational time [s] for each BAD's operation in seconds (orange column), contribution in percentage [%] for each BAD's operation (blue line) for Portugal case study

For completeness are reported the computational time also for the other two cases about the entire image:

- Spain: dimension 2929x2706 (rows x columns) with pixel size 10m; area considered 29,29x27,06 km;
- Chile: dimension 4587x4986 (rows x columns) with pixel size 10m; area considered 45,87x49,86 km:

	Spain (2929x2706)		Chile (4587x4986)	
	Time [s]	Contribution [%]	Time [s]	Contribution [%]
Compute Features	2,33	0,27	6,66	0,061
Compute MD	2,28	0,26	6,56	0,060
Compute OWA	24,20	2,76	194,56	1,773
Compute RG	848,11	96,60	10761,20	98,08
Compute Severity	0,89	0,10	2,66	0,024
Compute Severity+RG	0,09	0,01	0,20	0,002
TOT	877,90	100	10971,84	100

Table 5.16 Computational time for each BAD's operation for the entire study area in Spain (2929x2706 rows x columns; 29,29 km x 27,06 km) and Chile (4587 x 4986 rows x columns; 45,87 km x 49,86 km) expressed as total time [seconds] and percentage [%].



Charts 5.7 Computational time [s] for each BAD's operation (orange column), contribution in percentage [%] for each BAD's operation (blue line) for: the study area in Spain (a), Chile (b)

In order to run the algorithm for the test area in Spain (Chile), the total amount of time required by BAD plugin is equal to 14 minutes and 38 seconds (3 hours, 2 minutes and 51 seconds), and 96,6% (98,08%) of the time is required by the Region Growing step.

Another factor that could influence the amount of computational time required by BAD plugin is the number of initial seeds in the studied area, that, for the three test areas, are:

- Portugal:
    - initial number of seeds is equal to 447479;
    - final number of seeds, therefore pixels that are classified as Burned, is equal to 1310031;
- Spain:
    - initial number of seeds is equal to 390781, almost the 87% of the initial number of seeds in Portugal;
    - final number of seeds, is equal to 1917907, almost the 146% of the final number of seeds in Portugal;
- Chile:
    - initial number of seeds is equal to 1787526, almost the 400% of the initial number of seeds in Portugal and 457% with respect to Spain;
    - final number of seeds, is equal to 8131214, almost the 620% of the final number of seeds in Portugal and 424% with respect to Spain.

These initial number of seeds (pixels considered Burned) and so also the final ones, depends on:

- the Membership Function that is applied in order to compute the Membership Degree (MD);
- the OWA operator that is selected in order to integrated the information that come from the MD;
- the threshold that is set to generate the seed layer and the grow layer.

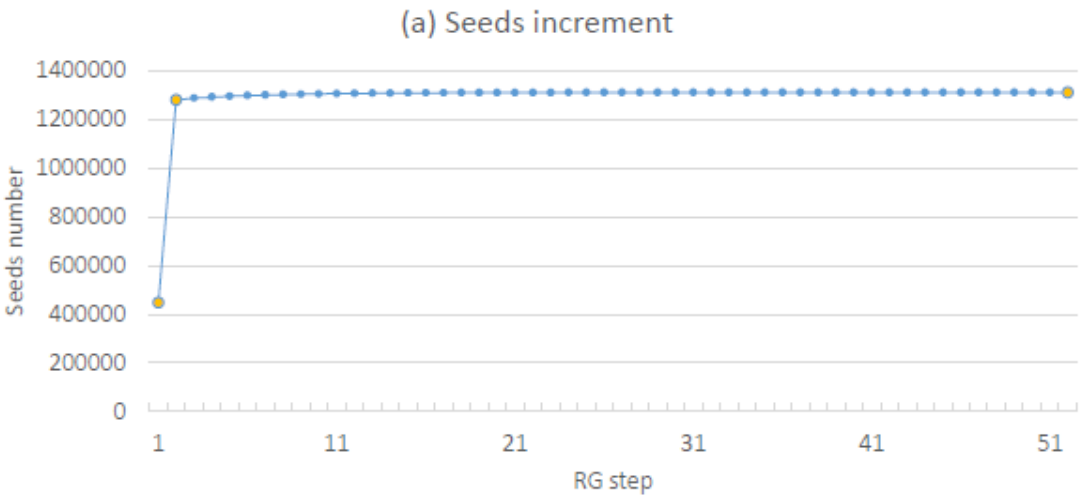
More the conditions considered are restrictive (optimistic vision of the event: OWA_AND as OWA operator, high value for the threshold), less will be the number of seeds.
In other hands, with a pessimistic approach to the event (OWA_OR as OWA operator, low value for the threshold), higher will be the number of seeds.
The number of seeds depends also on the impact and the extent of the wildfire event.

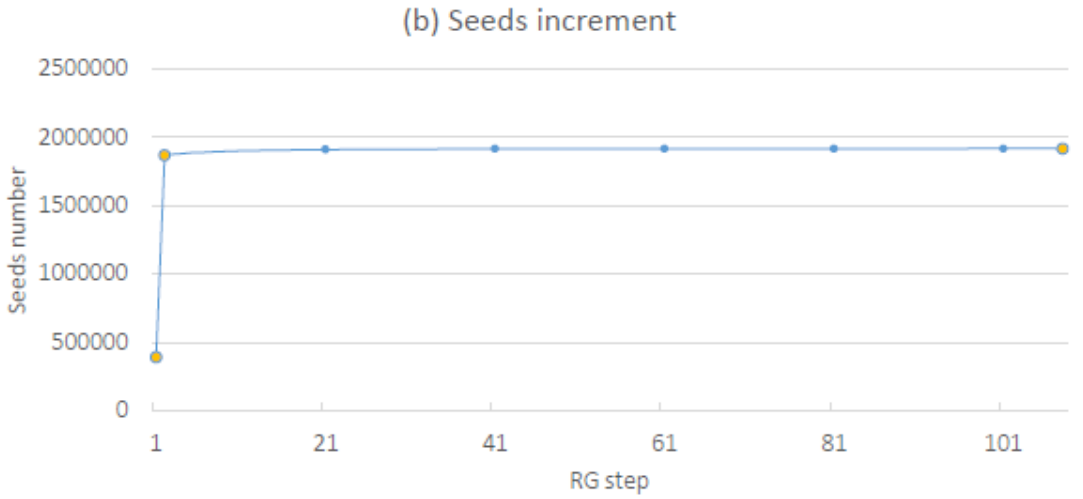
Another parameter that could quantify the different effort required in the RG step of the algorithm is the increase of the number of seed pixels after the first iteration and its distance from the final number of seeds:

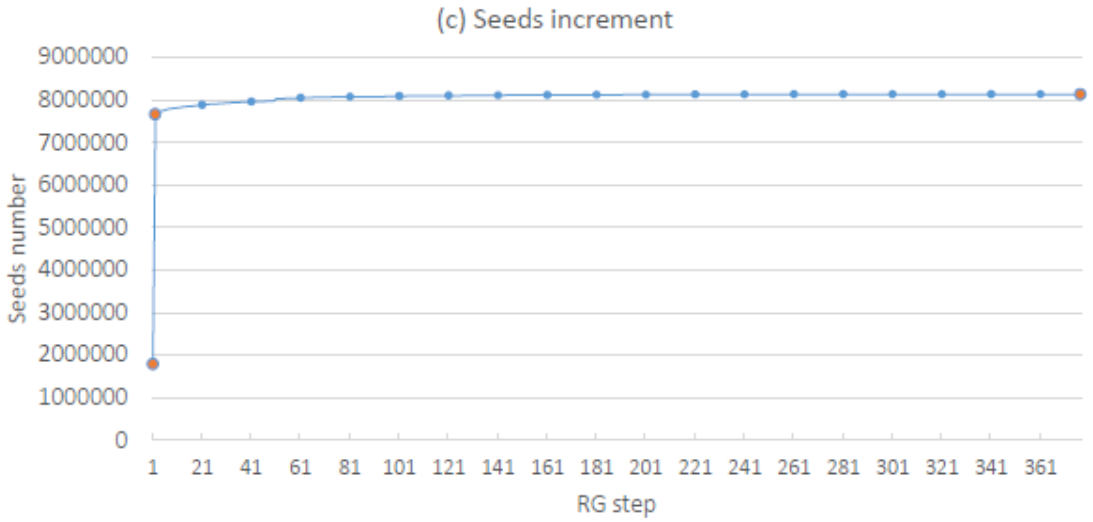
- Portugal: number of seeds after one iteration is equal to 1279562 that is the 97,6% of the final number of seeds. Number reaches after 52 steps;
- Spain: number of seeds after one iteration is equal to 1869109 that is the 97,45% of the final number of seeds. Number reaches after 108 steps;

- Chile: number of seeds after one iteration is equal to 7662363 that is the 94,23% of the final number of seeds. Number reaches after 377 steps.

This behavior is due to how the RG algorithm is implemented. During the application of the PixelByPixel methods, it immediately adds a new seed when a candidate, that meets the criterias, is found (paragraph 4.4.5.2). In this way, if the new element, that is just added, is on the right, or it is below with respect to the analyzed seed, it will be immediately considered as a new seed during the scrolling of the matrix rather than at the next iteration.







Charts 5.8 Seeds increment for Portugal (a), Spain (b), Chile (c) case studies (52 steps in RG phase); dots represent the number of seeds after each step, the orange dots are respectively: initial number of seeds, number of seeds after one iteration, final number of seed

6 CONCLUSION AND FUTURE PERSPECTIVES

The purpose of this thesis work was to implement the fuzzy burned area mapping algorithm developed by CNR-IREA [2] into a QGIS plugin to make available a tool for the identification of the areas affected by fires from classification of multi-spectral Sentinel-2 (S-2) images [4].

As described in Chapter 3, the algorithm takes as input two multi-spectral S-2 images collected: before and after the wildfire event.
These inputs are processed in order to produce a single output: Burned Area Map. This map is a binary classification product: 0 "Unburned", 1 "Burned".
Furthermore, the plugin integrates a burn severity algorithm that provides information on the level of damage induced on the vegetation by the fire. This algorithm relies on the difference of the Normalized Burn Ratio (NBR) index between pre-fire and post-fire images
([3]) and it takes as input the same S-2 images used for the fuzzy BA algorithm.
The main steps of the fuzzy burned area mapping algorithm, analyzed in detail in Chapter 3, are:

- computation and feature selection, algebraic operations are performed between image bands;
- use of fuzzy theory in order to calculate the membership degrees (MD), values that represent how much each image element/pixel belongs to the class under consideration ("Burned");
- use of the Ordered Weighted Averaging aggregation (OWA) operators for the purpose of integrating the previous information into a single synthetic value;
- application of the Region Growing algorithm.

In order to implement this algorithm, a plugin was developed in the GIS environment, Chapter 4: Burned Area Detector (BAD).
The plugin, developed in Python language, is installable and executable in QGIS.
The idea followed during the development of the plugin was to provide a simple, valid, effective and user-friendly tool for the classification of S-2 satellite images into burned/unburned.
The tool is easy-to-use by all types of users, from the most experienced to the least.

In order to achieve this goal, BAD presents itself, once installed and active in QGIS, with a practical graphical interface; the different tabs implement the steps of the fuzzy burned area mapping algorithm and the calculation of Burned Severity based on the dNBR algorithm ([3]). The output of the Burned Severity algorithm is combined with the result of the Burned Area algorithm to deliver a Burn Severity Map for the identified burned pixels.
Previous implementation of the fuzzy BA algorithm was available as R and Harris IDL scripts [2] that could limit its usability to more experienced users with a computer programming background. The interface developed in this thesis work and the availability within a GIS environment, makes the BAD plugin a tool available for a larger community of users.

Another objective of the BAD development is to leave the user as free as possible to customize some of the algorithm's steps and parameters. In particular the user can freely:

- provide the input images with the structure she/he desires;
- select the features that she/he wants;
- set the parameters that define the function for the Membership Degree calculation;
- select the OWA operators;
- select the thresholds for Region Growing;
- select the dNBR intervals for the Burn Severity classes.

In order to help the less experienced user, default settings are proposed for both the fuzzy BA and dNBR algorithms as a reference, in which the parameters and choices that will produce the best result are given, and can be modified if needed.

BAD plugin was tested on three different study areas: Portugal, Spain, Chile as reported in Chapter 5. The tests performed have mainly two objectives:

- the production of burned area and severity maps for three significant fire events. These maps are validated by comparison to reference datasets and accuracy metrics are estimated (section 5.3);
- the estimation of the time required to perform all the operations, i.e. the computational time (section 5.2).

For the purpose of validating the result, the following accuracy metrics are estimated, by comparing the results obtained through the BAD plugin with fire reference perimeters reported by the Copernicus EMS: omission error, commission error, Dice coefficient and relative bias.

Looking at the metric values obtained for the three case studies, it is possible to conclude that output BA maps are more than satisfactory with low commission and omission errors produced in the three scenarios, and the Dice coefficient very high: this means that the algorithm produces burned area maps that are in good agreement with the reference. Accuracy metrics obtained in the three test areas are consistent with values from the literature for regional applications (Sali et al. [2]).
However, for the Portugal and Spain study cases, burned area maps accuracy is significantly greater (DC > 94%). An exception to this trend is the study case of Chile where commission error is greater (~15%) due to an area burned before the considered event, and so detected by the BAD plugin as a new burned area. The reason is the features and the OWA operator selected during the application of the plugin.

Regarding the second objective, it is evident that satellite images in raster format can cover larger areas hence they might take a long time to process: for larger areas, processing time could be even on the order of hours.
In the fuzzy BA algorithm, the Region Growing step is the operation that takes more time than others. This happens because Region Growing is an iterative and contextual algorithm and at each iteration each pixel, and its neighbors, must be examined. However, the RG is necessary to reduce commission errors by exploiting the contextual information for the burned pixels (new burned pixels are added only if contiguous to burned pixels from the previous iteration) by at the same time reducing omission over low severity burned pixels. Indeed, the RG algorithm has been largely proved as a suitable approach for balancing omission and commission errors in burned area mapping algorithms (Sali et al. [2], Bastarrika et al. [58]).

From a future perspective, multiple aspects could be considered for the purpose of improving and expanding the plugin's functionalities:

- allow the use of input images from other satellite missions (i.e., different number of bands). However, the more experienced user could already use BAD as long as she/he does not exceed the maximum allowed size of the multi band input raster (BAD accepts a maximum of 13 bands);
- allow the user to use Features and/or OWA as desired;
- flexible calibration of the Membership Function parameters according to the input, thus according to the selected geographic area;
- implement a part of the plugin to handle masks to be applied to the input;
- handle errors made by the user, through error messages produced by the plugin;
- optimize the Region Growing implementation, in order to reduce computational time;

- enable the plugin to locate and collect data directly from a Web Service, so that the user does not have to search and download images of the selected area each time, saving memory in her/his device.
- implement a validation module to produce directly via plugin a comparison with reference data.

All these aspects would improve the BAD plugin and its performance, ensuring the validity of the results: a greater variety and accessibility of input data, reduced computation time, local memory savings, and the ability to analyze areas of any size in a short time, by using accessible tools and devices.

Bibliography & Sitography

[1] «ESA, Copernicus, Sentinel-2,» [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2.

[2] Sali, Piaser, Boschetti, Brivio, Sona, Bordogna e Stroppiana, «A Burned Area Mapping Algorithm for Sentinel-2 Data Based on Approximate Reasoning and Region Growing,» 2021. [Online]. Available: https://www.mdpi.com/2072-4292/13/11/2214.

[3] N. B. Carl H Key, «Landscape Assessment (LA) Sampling and Analysis Methods,» in *FIREMON: Fire Effects Monitoring and Inventory System*, 2005, pp. 1-51.

[4] T. Martinoli, «BAD_QGISplugin,» [Online]. Available: https://github.com/ThomasMartinoli/BAD_QGISplugin.git.

[5] «Wildifire, Copernicus,» [Online]. Available: https://climate.copernicus.eu/esotc/2021/wildfires.

[6] «Mediterranean summer extremes, Copernicus,» [Online]. Available: https://climate.copernicus.eu/esotc/2021/mediterranean-summer-extremes .

[7] Rogers, Balch, Goetz, Lehmann e Turetsky, «Focus on changing fire regimes: interactions with climate, ecosystems, and society,» 6 March 2020. [Online]. Available: https://iopscience.iop.org/article/10.1088/1748-9326/ab6d3a.

[8] NASA, «Six trends to know about fire season in the western U.S,» [Online]. Available: https://climate.nasa.gov/ask-nasa-climate/2830/six-trends-to-know-about-fire-season-in-the-western-us/.

[9] R. Morrison, «Earth ORG "The environmental Impact of Wildfires",» 18 August 2022. [Online]. Available: https://earth.org/environmental-impact-of-wildfires/.

[10] W. H. O. WHO. [Online]. Available: https://www.who.int/health-topics/wildfires#tab=tab_1 .

[11] Copernicus, «About Copernicus,» [Online]. Available: https://www.copernicus.eu/en/about-copernicus.

[12] ESA, «The Sentinel missions,» [Online]. Available: https://www.esa.int/Applications/Observing_the_Earth/Copernicus/The_Sentinel_missions.

[13] ESA, «Satellite Description,» [Online]. Available: https://sentinel.esa.int/web/sentinel/missions/sentinel-2/satellite-description.

[14] Copernicus, «Copernicus Open Access Hub,» [Online]. Available: https://scihub.copernicus.eu/.

[15] ESA, «Data Products,» [Online]. Available: https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-2/data-products.

[16] ESRI, «What is GIS?,» [Online]. Available: https://www.esri.com/en-us/what-is-gis/overview.

[17] «QGIS,» [Online]. Available: https://www.qgis.org/it/site/.

[18] T. Sutton, «[Qgis-developer] Announcing the release of QGIS 1.0 'Kore',» January 2009. [Online]. Available: https://lists.osgeo.org/pipermail/qgis-developer/2009-January/005774.html.

[19] D. Kuhlman, in *A Python Book: Beginning Python, Advanced Python, and Python Exercises*, 2012, p. Section 1.1.

[20] «Zen of Python,» [Online]. Available: https://en.wikipedia.org/wiki/Zen_of_Python.

[21] L. B.-C. a. J. S.-M.-A. Brigitte Leblon, «Use of Remote Sensing in Wildfire Management,» 2012. [Online]. Available: https://www.intechopen.com/chapters/38093.

[22] «Dramatic photos capture California's record-breaking fires,» [Online]. Available: https://www.latimes.com/story/2021-07-12/see-these-incredible-photos-as-california-hit-by-record-breaking-fire.

[23] «Climate change is here, and the world is burning,» [Online]. Available: https://www.theglobeandmail.com/opinion/article-climate-change-is-here-and-the-world-is-burning/].

[24] Copernicus, «Active Fire Detection,» [Online]. Available: https://effis.jrc.ec.europa.eu/about-effis/technical-background/active-fire-detection.

[25] J. D. C. O. J. Y. J. K. Louis Giglio, «An Enhanced Contextual Fire Detection Algorithm for MODIS,» *Remote Sensing of Environment,* pp. 273-282, 2003.

[26] N. EARTHDATA, «MCD14DL-NRT,» [Online]. Available: https://www.earthdata.nasa.gov/learn/find-data/near-real-time/firms/mcd14dl-nrt.

[27] N. EARTHDATA, «VIIRS I-Band 375 m Active Fire Data,» [Online]. Available: https://earthdata.nasa.gov/earth-observation-data/near-real-time/firms/viirs-i-band-active-fire-data.

[28] NASA, «Fire Information for Resource Management System,» [Online]. Available: https://firms.modaps.eosdis.nasa.gov/map/#d:24hrs;@0.0,0.0,3z.

[29] M. Bucher, «Forest fires rage across the western US, how vulnerable is Pennsylvania to fire risk?,» [Online]. Available: https://www.witf.org/2021/07/28/forest-fires-rage-across-the-western-us-how-vulnerable-is-pennsylvania-to-fire-risk.

[30] F. M. G. R. V. D. W. e. a. Emilio Chuvieco, «Historical background and current developments for mapping burned area from satellite Earth observation,» *Remote Sensing of Environment,* pp. 45-64, 2019.

[31] L. B. Andrea Melchiorre, «Global Analysis of Burned Area Persistence Time with MODIS Data,» *Remote Sensing,* 2018.

[32] G. M. M. K. N. Koutsias, «A forward/backward principal component analysis of Landsat-7 ETM+ data to enhance the spectral signal of burnt surfaces,» *Remote Sensing,* pp. 37-46, 2009.

[33] J. Richards, «Thematic mapping from multitemporal image data using the principal components transformation,» *Remote Sensing of Environment,* pp. 35-46, 1984.

[34] A. S. M.C. Pereira, «Spectral characteristics of deforestation fires in NOAA-AVHRR images,» *International ournal of Remote Sensing,* pp. 583-597, 1993.

[35] M. K. N. Koutsias, «Logistic regression modelling of multitemporal Thematic Mapper data for burned area mapping,» *International Journal of Remote Sensing,* pp. 3499-3514, 1998.

[36] M. K. N. Koutsias, «Burned area mapping using logistic regression modeling of a single post-fire Landsat-5 Thematic Mapper image,» *International Journal of Remote Sensing,* pp. 673-687, 2000.

[37] M. G. D. R. E. C. R. Ramo, «A data mining approach for global burned area mapping,» *International Journal of Applied Earth Observation and Geoinformation,* pp. 36-51, 2018.

[38] G. B. P. C. M. B. L. B. P. B. D. Stroppiana, «A method for extracting burned areas from Landsat TM/ETM+ images by soft aggregation of multiple Spectral Indices and a region growing algorithm,» *ISPRS Journal of Photogrammetry and Remote Sensing,* pp. 88-102, 2012.

[39] «dzetsaka : Classification tool,» [Online]. Available: https://plugins.qgis.org/plugins/dzetsaka/.

[40] J. E. Keely, «Fire intensity, fire severity and burn severity: A brief review and suggested usage,» *International Journal of Wildland Fire,* pp. 116-126, 2009.

[41] U. Nations, «Normalized Burn Ratio (NBR),» [Online]. Available: https://un-spider.org/advisory-support/recommended-practices/recommended-practice-burn-severity/in-detail/normalized-burn-ratio.

[42] «QGIS Fire Mapping Tool,» [Online]. Available: https://www.mtbs.gov/qgis-fire-mapping-tool.

[43] Copernicus, «Emergency,» [Online]. Available: https://www.copernicus.eu/en/copernicus-services/emergency.

[44] Roteta, Bastarrika, Padilla, Storm e Chuvieco, «Development of a sentinel-2 burned area algorithm: Generation of a small fire database for sub-Saharan Africa,» *Remote Sensing of Environment,* pp. 1-17, 2019.

[45] R. Yager, «Quantifier Guided Aggregation Using OWA Operators,» *International Journal of Intelligent Systems,* pp. 49-73, 1996.

[46] C. D. U. Darwin Centre for Bushfire Research, «The NAFI Booklet,» 2016. [Online]. Available: https://www.google.com/url?q=https://firenorth.org.au/nafi3/views/help/NAFI_Booklet.pdf&sa=D&source=docs&ust=1680974360483112&usg=AOvVaw1DWAkZXiS1l32Tgtw2rAx3.

[47] «NAFI Frequently Asked Questions,» 2010. [Online]. Available: https://i.unu.edu/media/tfm.unu.edu/page/389/Using-the-NAFI-Site.pdf.

[48] L. o. F. M. a. R. Sensing, «NOFFi Burned Area Mapping,» [Online]. Available: http://epadap.web.auth.gr/?page_id=2175&lang=en.

[49] S. o. F. a. N. E. A. U. o. T. Laboratory of Forest Management and Remote Sensing, «The Greek National Observatory of Forest Fires (NOFFi),» 2016.

[50] «Plugin Builder 3,» [Online]. Available: https://plugins.qgis.org/plugins/pluginbuilder3/.

[51] «Qt Documentation,» [Online]. Available: https://doc.qt.io/.

[52] «PyQt5,» [Online]. Available: https://pypi.org/project/PyQt5/.

[53] C. EMS, «LATEST NEWS ·2023-03-13 | [EMSR654] Tropical Cyclone Freddy in Mozambique,» [Online]. Available: https://emergency.copernicus.eu/mapping/list-of-components/EMSR207/ALL/EMSR207_05PEDROGAOGRANDE.

[54] C. EMS, «EMSR625: Wildfire in the Valencian Community, Spain,» [Online]. Available: https://emergency.copernicus.eu/mapping/list-of-components/EMSR625.

[55] C. EMS, «EMSR647: Forest Fires in Chile,» [Online]. Available: https://emergency.copernicus.eu/mapping/list-of-components/EMSR647.

[56] R. a. G. K. Congalton, «Assessing the Accuracy of Remotely Sensed Data Principles and Practices,» 1999.

[57] J. L.-L. S. V. S. E. C. Magí Franquesa, «Using long temporal reference units to assess the spatial accuracy of global satellite-derived burned area products,» *Remote Sensing of Environment,* 2022.

[58] E. C. M. P. M. Aitor Bastarrika, «Mapping burned areas from Landsat TM/ETM+ data with a two-phase algorithm: Balancing omission and commission errors,» *Remote Sensing of Environment,* pp. 1003-1012, 2011.

List of Figures

List of Tables

List of Charts

List of Formulas

List of Scripts