POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA CIVILE,
AMBIENTALE E TERRITORIALE

Master of science in Geoinformatics Engineering
Dipartimento di Ingegneria Civile e Ambientale

Landslide susceptibility mapping and displacement monitoring: a case study in Northern Italy

Supervisor: Prof. Maria Antonia Brovelli
Co-supervisor: Dr. Vasil Yordanov

Thesis by: Lorenzo Amici
Student ID: 927524

Academic year 2020 - 2021

Contents

List of Figures

List of Tables

Listings

Abstract

Landslides are one of the most dangerous and disastrous geological hazard worldwide, posing threats to human life, infrastructures and to the natural environment. This dissertation aims to analyse these phenomena both at the basin scale, by producing landslide susceptibility maps, and at a single landslide scale, by monitoring its displacements through satellite images.

Landslide susceptibility mapping is a topic of crucial importance in risk mitigation. In this work, a machine learning approach based on the Random Forests algorithm is adopted to produce landslide susceptibility maps over two areas in Northern Lombardy (Val Tartano and Upper Valtellina), Italy. Following a state of the art analysis on this topic, the Random Forests technique was chosen for its positive performances, that were further confirmed by this work. An innovative aspect of this dissertation is the introduction of a *No Landslide zone* defined by geological criteria, which aims to determine areas with very low possibility of landslides. By these means, the model was provided with information about landslide absence in addition to that of past landslide events. The models obtained were subsequently validated with state-of-the-art metrics, showing satisfactory results.

Whilst susceptibility studies can be of great aid in preventing threats posed by future events, active landslides need to be monitored to reduce the risk of damages and casualties. With this aim, this work proposes a way to compute landslide displacements through time, by exploiting the great availability of high quality multispectral satellite images. The developed procedure produces maps of displacement magnitude and direction by means of local cross-correlation of Sentinel-2 images.

The Ruinon landslide, an active landslide in Upper Valtellina, was analysed during two different time windows.

Both the analyses described in this work were designed to be entirely based on free and open-source GIS software and to rely exclusively on open data. These characteristics allow the proposed analyses to be easily replicated, customized, and empowered.

Sommario

Le frane sono uno dei fenomeni naturali più pericolosi e disastrosi a livello globale, e rappresentano una minaccia per gli uomini, le infrastrutture e per l'ambiente naturale. Questa tesi ha quindi l'obiettivo di analizzare tali fenomeni sia alla scala di bacino, producendo mappe di suscettibilità da frana, sia concentrandosi su singole frane, monitorandone gli spostamenti attraverso immagini satellitari.

La mappatura di suscettibilità da frana è un argomento di fondamentale importanza nel campo della mitigazione del rischio. In questo lavoro, è stato adottato un approccio di machine learning basato sull'algoritmo Random Forests per la produzione di mappe di suscettibilità da frana in due aree nel nord della Lombardia (Val Tartano e Alta Valtellina), in Italia. Dopo un'analisi dello stato dell'arte in questo campo, la tecnica Random Forests è stata selezionata per le sue prestazioni positive, che sono state inoltre confermate in questo lavoro. Un aspetto innovativo di questa tesi è l'introduzione di una *No Landslide zone* definita tramite criteri geologici, che ha lo scopo di identificare zone in cui il rischio di frane è molto ridotto. Così facendo, al modello sono state fornite informazioni riguardo l'assenza di frane in aggiunta a quelle di eventi franosi passati. I modelli così ottenuti sono stati validati con metriche all'avanguardia, producendo risultati soddisfacenti.

Mentre gli studi di suscettibilità possono essere di grande aiuto nel prevenire i pericoli di eventi futuri, le frane già attive devono essere monitorate con lo scopo di ridurre il rischio di danni. A tal fine, questo lavoro propone una procedura per calcolare gli spostamenti di frane nel tempo, facendo leva sull'ampia disponibilità di immagini satellitari multispettrali di alta qualità. La procedura sviluppata produce come

risultato mappe della lunghezza e della direzione dello spostamento usando una cross-correlazione locale di immagini Sentinel-2. La frana del Ruinon, una frana attiva in Alta Valtellina, è stata analizzata in due finestre temporali distinte.

Entrambe le analisi descritte in questa tesi sono state progettate con l'intento di essere completamente basate su software GIS free e open-source e di utilizzare esclusivamente dati open. Tali caratteristiche rendono queste analisi facilmente replicabili, personalizzabili e migliorabili.

Chapter 1

Introduction

Landslides are one of the most dangerous and disastrous geological hazard worldwide (Guzzetti et al., 1999; Reichenbach et al., 2018); their occurrence is the cause for economic losses, casualties and damage to the natural environment and to human infrastructures (Guzzetti et al., 2012; Aditian et al., 2018). It has also been shown that the population growth and the consequent expansion of settlements of the last decades are magnifying the impact of such phenomena (Guzzetti et al., 2012). Moreover, we are also witnessing a rise in the extreme events that trigger landslides, as a consequence of climate change (Machichi et al., 2020).

Therefore, landslide susceptibility mapping, i.e. the estimation of the likelihood of landslide events in a territory based on environmental conditions, is a subject of primary importance. Its functioning principles and assumptions are, according to Guzzetti et al. (1999):

- Future landslides are more likely to occur under the conditions which led to past and present instabilities. Hence, past failures are the starting point for landslide susceptibility mapping;
- The landsliding process is regulated by mechanical laws, and the factors that lead to slope failures can be collected and used to build predictive models of landslide occurrence;
- Slope failures are easily recognizable because they give rise to distinguishable morphological features;

- The probability of occurrence of landslides can be inferred from heuristic investigations, computed with the analysis of environmental information, or even derived from physical models. Therefore, a territory can be divided into hazard classes, according to different probabilities thresholds.

A wide variety of approaches have been adopted by scholars in the past to produce susceptibility maps; in general, they belong to two different families (Reichenbach et al., 2018; Zhou et al., 2018):

- Qualitative methods: subjective, inventory-based and knowledge driven approaches, that describe the susceptibility status of a given area using descriptive terms;
- Quantitative methods: data-driven and physically-based approaches, that return probabilistic values in order to characterize the susceptibility level of an area.

In a literature review published in 2018, Reichenbach et al. summarized and classified the methods for producing landslide susceptibility maps, dividing them in five main categories:

- Geomorphological mapping: a geological professional directly assesses the susceptibility level;
- Analysis of landslide inventories: investigation of past landslide density maps is used in order to predict future events;
- Heuristic or index-based approaches: investigators rank a set of instability factors based on their relevance in causing landslides;
- Process based methods: a simplified physical scheme modelling landslides is used to analyse stability/instability conditions;
- Statistically-based modelling methods: approach based on the statistical analysis and combination of a set of instability factors and the past or present distribution of landslide events.

On the other hand, monitoring already active landslide is an equally important subject for mitigating the risk posed by these natural phenomena. Furthermore, in the last years we have witnessed a huge increase in the availability of free and open multispectral, multitemporal and global coverage satellite imagery; at the same time, also new open software tools for exploiting these images have arisen. Therefore

another part of this study focuses on the analysis of satellite imagery using free and open source GIS software to identify displacements of single landslides.

The aim of the work outlined in this thesis is to implement a statistical-based method (Random Forests) to produce Landslide Susceptibility maps at the basin level in two zones in Northern Lombardy, and to validate the obtained results. Secondly, this work focuses on creating a semi-automatic landslide movement detection procedure that can be easily replicated and customized, given the large availability of short-revisiting time open satellite images and since it employs only open software.

1.1 Thesis outline

The thesis is structured in the following way:

Chapter 1 introduces the work.

Chapter 2 outlines the current state of the art regarding landslide susceptibility analysis and landslide displacement monitoring, and the choices of the employed approaches are outlined and explained, taking into consideration previous studies.

Chapter 3 describes the selected areas of interest and their geomorphological features, in order to better frame the context of the study. On top of that, it contains the description of the data collected and of the preparation steps applied to the dataset.

Chapter 4 illustrates the software tools, providing a description and the purpose of each one of them, and also presents the analysis techniques required by this study, diving into a theoretical explanation about their functioning.

Chapter 5 contains an in-depth description of the work carried out in this thesis, illustrating the analysis process and the various modifications introduced during the work.

Chapter 6 presents the obtained results and the applied validation processes.

Chapter 7 includes the general conclusions derived from the work.

Chapter 2

State of the art

2.1 Landslide susceptibility mapping

Statistically-based landslide susceptibility mapping consists in the construction of a statistical model, with the purpose of relating geo-environmental factors with a set of known landslide events; this model can be exploited to forecast susceptibility levels in a given territory.

The first studies on landslide susceptibility modelling using statistical approaches were done in the late 70s and early 80s (Neuland, 1976; Carrara, 1983). The interest in these types of analyses has grown over the years, supported by the growth of more complex and precise analytical techniques.

Reichenbach et al. in 2018 reviewed researches in the field of landslide susceptibility from 1983 to 2016. In this review, they grouped the various statistical techniques used for susceptibility assessment; here are reported the six main groups to which the identified methods belong:

- Classical statistics (e.g. logistic regression, discriminant analysis, linear regression): these approaches generally rely on having an explicit underlying probability model (Fulkerson et al., 1995), and often compute the probability of being in each of the classification classes. It is common for these methods to require human intervention when selecting and transforming the variables, and in structuring the problem;

- Index-based (e.g. weight-of-evidence, heuristic analysis): in this case, the presence of professional figures is key for evaluating the susceptibility level based on a set of instability factors or on environmental indexes;
- Machine learning (e.g. fuzzy logic systems, support vector machines, forest trees): the term Machine Learning usually describes automatic computing procedures that learn a task from a series of examples. When talking about classification problems, scholars have focused their attention on decision trees approaches, in which classification results from a series of logical or binary operations (Fulkerson et al., 1995). In general, machine learning techniques are able to represent more complex problems with respect to classical statistics, and the operational steps do not require human intervention;
- Neural networks: these techniques were first developed with the intent of emulating the functioning of the human brain. To do so, they usually consists of layers of interconnected nodes, where each node outputs a non-linear function of its input. The whole network therefore represents *"a very complex set of interdependencies which may incorporate any degree of nonlinearity, allowing very general functions to be modelled"* (Fulkerson et al., 1995);
- Multi Criteria Decision Analysis: GIS-based MCDA is a process that transforms and combines geographical data and value judgements to obtain information for decision making (Malczewski, 1999);
- Other statistics (Data Overlay Analysis, etc.).

The review also shows that in the period 1983-2016 logistic regression has been by far the most commonly used method for Landslide Susceptibility Mapping (LSM).

2.1.1 Workflow

Literature analysis shows a common workflow for the creation of landslide susceptibility maps, that can be summarized in five steps:

- data selection, that comprises:
  - selection of the landslide inventory to be used in the analysis. The most common approach is to use a single inventory, but there are also articles including multiple inventories or multi-temporal ones;

  - selection of the environmental factors necessary to build the model. A broad categorization of the factors employed in the literature could be: geological, hydrological, land cover, morphological and other types of variables (Reichenbach et al., 2018);

- division of the territory in mapping units, i.e. a *"portion of the land surface which contains a set of ground conditions which differ from the adjacent units across definable boundaries"* (Guzzetti et al., 1999). The appropriate mapping unit should preserve internal unit homogeneity while highlighting between-unity heterogeneity (Reichenbach et al., 2018). The most common choice by far is to use grid cells, i.e. pixels, as mapping unit;
- division of the landslide inventory in a training set (the part that will be fed to the model in order to create the map) and a test set (the part that will be used for testing the obtained results);
- application of the chosen model, thus obtaining the susceptibility map;
- validation of the model using the chosen methods. There are many indices and metrics to evaluate the performances of the model (Guzzetti et al., 2006; Rossi et al., 2010; Reichenbach et al., 2018), but it is crucial to differentiate between the evaluation of the model fit and of the model performance. The evaluation of the model fit describes the ability of the model to correctly classify the data that has been used to build the model, i.e. we can obtain the model fit by comparing the model outcomes with the training dataset. On the other hand, the model predictive performance indicates the capability of the model to correctly predict a high susceptibility level where landslides unknown to the model already exist and a low susceptibility level where landslides are unlikely to happen; this can be obtained by comparing the model against the testing dataset.

2.1.2 Review of machine learning methods for susceptibility mapping

This thesis work focuses on the application of Machine Learning (ML) methods to the landslide susceptibility mapping operation. Machine learning analytic tools aim to build a model able to represent relationships, that beforehand are totally or

partially unknown, between data and target variables (Ma et al., 2020). Advantages of ML approaches with respect to classical statistical methods are the reduced amount of data required by ML techniques for obtaining reliable results, and the fact that no statistical assumption is necessary to build ML models (Lee et al., 2003; Pourghasemi and Rahmati, 2018).

In the last years, machine learning has been widely used for Landslide Susceptibility assessment analyses, and the vast majority of the employed methods fall under the umbrella of *supervised learning*, meaning that the aim is to build a connection between known inputs and unknown outputs. This is because Landslide Susceptibility studies often make use of the assumption that landslides are more likely to occur in conditions similar to those that caused landslide events in the past (Prakash et al., 2020). In general, *supervised learning* can be divided into two categories: classification, where the desired output consists of a set of classes or labels, and regression, where the aim is to predict a continuous variable (Ma et al., 2020). Both classification and regression can be applied in landslide susceptibility problems: the purpose of the former is usually to divide the territory into "landslide" and "no landslide" classes, while the latter computes a probability value associated with the risk of mass movement events.

In the rest of this chapter, the decisive aspects of a machine learning landslide susceptibility analysis are described, and different approaches between previous studies are outlined.

2.1.2.1 Employed model

Some of the most applied ML algorithms in Landslide Susceptibility Mapping are: Logistic Regression, Support Vector Machines, Random Forests, fuzzy logic systems, decision trees, Artificial Neural Networks (Goetz et al., 2015; Pham et al., 2016; Tien Bui et al., 2016; Pourghasemi and Rahmati, 2018).

Logistic Regression (LR) seems to be by far the most common method, and is followed by Support Vector Machine (SVM) in terms of popularity (Reichenbach et al., 2018).

Methods like LR and SVM can be considered *simple ML* algorithms; in fact, they were often employed in the early days of ML approaches to Landslide Susceptibility

problems (Lee and Sambath, 2006; Yao et al., 2008). Artificial Neural Networks (ANN) and decision trees can also be considered *simple ML* algorithms, and were applied as well in Landslide Susceptibility analyses (Lee et al., 2003; Saito et al., 2009).

Over the last years, researchers in this topic seem to be more inclined to consider more complex algorithms (Yordanov et al., 2021), like Random Forests (RF), complex neural network structures like Convolutional Neural Networks (CNN) or Recurrent Neural Network (RNN) and ensemble learning methods (Wang et al., 2019; Thi Ngo et al., 2021). From the theoretical point of view, these complex methods usually arise or are built starting from simpler methods. Because of this, these new, more complex methods usually generate more robust landslide models than the base classifiers (Fang et al., 2021).

In the particular case of this work, the Random Forests algorithm has been employed, since it has been already widely used in the field of landslide susceptibility analysis, and it has been proven to have good overall performances (Catani et al., 2013; Pourghasemi and Rahmati, 2018; Dou et al., 2019; Emami et al., 2020; Yordanov and Brovelli, 2020a). Moreover, studies that analysed multiple methods comparing their performances often found RF to be the best model in terms of predictive capability. For example, Pourghasemi and Rahmati (2018) compared the performance of ten different machine learning techniques, including RF, ANN and SVM, in an area of about 2241 km^2 in Iran, using the area under the ROC curve (AUC-ROC) approach for evaluating the models' performances; the research found that RF and Boosted Regression Trees, another advanced ML technique, outperformed the other methods. Similar results were obtained by other recent studies, like Dou et al. (2019), Emami et al. (2020) and Yordanov and Brovelli (2020b). Goetz et al. in 2015 studied the differences between machine learning algorithms, including RF, and conventional statistical prediction techniques (i.e. weight-of-evidence and generalized additive models) and proved that RF had the overall best predictive performances.

One of the advantages of the Random Forests technique is being an ensemble learning method, since it is composed by multiple decision trees working together. When using only one decision tree the random selection of the training dataset could affect the results, but RF overcomes this issue by using a set of many trees in order to ensure the stability of the model (Ma et al., 2020).

2.1.2.2 Chosen factors

The selection of the environmental variables to consider in the analysis is a fundamental step for susceptibility mapping with statistical methods, and in the literature we find a broad variety of factors combinations. The choice of the environmental variables to include in a landslide susceptibility analysis is highly influenced by the chosen technique, the study area, the landslide types and the availability of data, and usually depends on decisions made by experts. Because of these reasons, it is not easy to standardize what could be the best factors set overall. What could be done instead is analysing the literature in order to find which are the terrain variables that are more often factored in the research; the review made for writing this thesis showed that some of the most common factors taken into consideration are:

- slope;
- aspect;
- lithology;
- altitude;
- curvature:
- distance from faults;
- distance from rivers;
- distance from roads;
- land use or land cover.

2.1.2.3 Validation metrics

A crucial step of any modelling process, and in particular of Landslide Susceptibility mapping, is the validation of the model. Despite the importance of this task, usually scholars are more focused on implementing new and more complex models in order to obtain better results, rather than thoroughly evaluating the produced model. In a literature review, Reichenbach et al. (2018) highlighted that many studies did not implement any method to validate the predictive or fitting capability of the obtained models (38.9% and 32.0% respectively), and that almost half of them were relatively recent studies (published between 2010 and 2016). On the other hand, the

review also showed an increase in the number of validation methods available, and identified the most popular ones in the success/prediction rate curves (Chung and Fabbri, 1999, 2003) and Receiver Operating Characteristic (ROC) curves (Ayalew and Yamagishi, 2005). In a more recent article, Yordanov and Brovelli (2020b) compared the Area Under the Curve ROC method (AUC-ROC) with the Precision Recall Curve (PRC) in the context of Landslide Susceptibility Mapping. They showed that, when working with an imbalanced dataset, i.e. a dataset in which there is a difference between the number of positive and negative cases in a binary classification (Saito and Rehmsmeier, 2015), the PRC outlines the performances of the model more precisely, while the two methods have similar behaviours when the dataset is balanced.

2.2 Landslide displacement monitoring

Satellite images have been employed by scholars for monitoring various phenomena on the surface of the Earth: glacier movement (Berthier et al., 2005), land cover changes (Dewan and Yamaguchi, 2009), forest growth or deforestation (Coppin and Bauer, 1996; Kennedy et al., 2010), landslides (Colesanti and Wasowski, 2006), and others.

In this work, landslide displacements through time are monitored by means of a Maximum Cross-Correlation (MCC) method. In the past this technique was mainly applied to geophysical phenomena involving fluid motion, such as the movement of clouds (Leese et al., 1971), sea (Crocker et al., 2007) or glaciers (Ninnis et al., 1986).

Concerning terrestrial land-cover changes, techniques based on cross-correlation have been generally applied by comparing pixels or objects in two different images; these methods usually produce maps that indicate if a pixel value has changed or not between the two images, without being able to identify a movement vector. On the other hand, more recent studies have applied the MCC method to identify directional changes, quantifying the movement detected for every single pixel (You et al., 2017; Oxoli et al., 2020).

You et al. (2017) were the first to apply MCC for detecting terrestrial land-cover changes. In their study they applied the MCC method to identify and quantify the direction of land-cover changes in a river floodplain in Bolivia using Landsat-5

Thematic Mapper images. The results obtained confirmed the validity of Maximum Cross-Correlation to quantitatively identify changes between satellite images.

Oxoli et al. (2020) considered Landsat-8 and Sentinel-2 satellite images to detect the movements of desert dunes using MCC. Their analysis concludes that the displacements identified by the procedure are slightly overestimated with respect to reference data, while the direction of the movements generally agrees with environmental factors such as wind direction and morphology.

Chapter 3

Areas of interest and data

3.1 Areas of interest

Before carrying out the analyses, it was important to define the boundaries of the areas of interest (AOIs) for this study (Figure 3.1). Lombardy region is heavily affected by hydro-geological natural hazards, particularly in pre-Alpine and Alpine areas, where the combination of heavy rainfall and snowmelt often creates hydrogeological conditions that lead to the occurrence of shallow landslides. Therefore, this study focuses on two areas in Lombardy with a particularly great abundance and variety of landslide phenomena.

3.1.1 Val Tartano

The first area analysed by this study is the basin of Val Tartano. Val Tartano is located in the Lombardy region, Northern Italy; the basin extends for 51 km^2 and is characterized by steep slopes and an elevation ranging from 250 to 2500 m a.s.l. From the geological point of view, the valley contains numerous faults, mostly with NE-SW and NW-SE strikes, accompanied by shear zones (Longoni et al., 2016). Coupling these features with the river network makes the area prone to instabilities and landslide phenomena of various types. In fact, the landslide inventory of ISPRA (Istituto Superiore per la Protezione e la Ricerca Ambientale) shows more than 1000 single landslide events in the area considered.

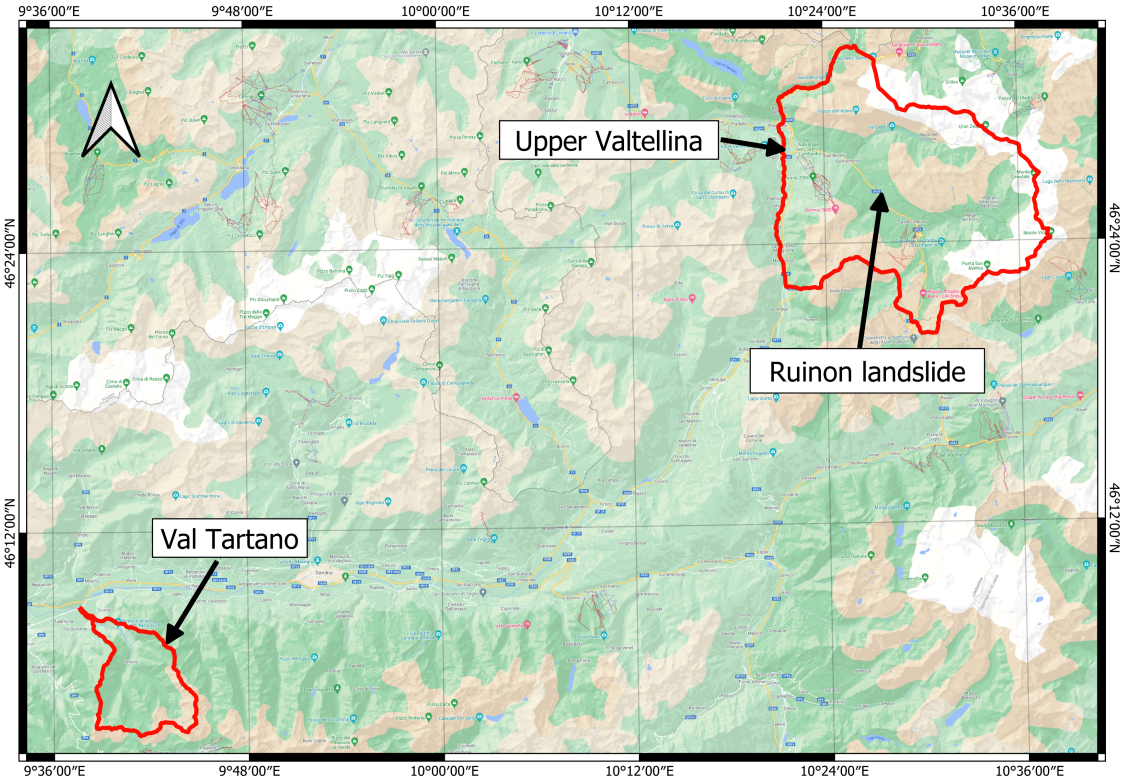


Figure 3.1: The two areas of interest: Tartano basin (on the left) and Upper Valtellina (on the right)

3.1.2 Upper Valtellina

The other area of interest for this study is the area of Upper Valtellina, also located in Lombardy, near the border with Trentino - Alto Adige. The study area covers about 295 km^2, and is characterized by steep slopes and an elevation ranging from 900 to 3800 m a.s.l. Due to the high altitude of the area, it is common that instabilities arise because of the decompression phenomena due to deglaciation processes. For this area, the ISPRA landslide inventory contains 3644 single landslide events, including 11 Deep-seated Gravitational Slope Deformations.

3.1.3 Ruinon landslide

The monitoring part of this thesis focuses on one particular landslide: the Ruinon landslide (Figure 3.2), situated in Upper Valtellina. This landslide is one of the most active landslides in the Alps, and it is believed to extend down to a depth of 50–70

m, for a total estimated volume of approximately 30 million m^3. The landslide is situated at the base of a Deep-seated Gravitational Slope Deformation, that affects the entire slope up to the summit at 3000 m a.s.l. Two major scarps can be identified: the upper one is a sub-vertical rock cliff of about 30 m in height, while the lower one is characterized by a more widespread debris cover. Over the years, a large lobe of chaotic debris has propagated towards the valley bottom, giving origin to secondary mass wasting processes in the form of rockfalls, debris flows, and shallow slumps (Carlà et al., 2021).

Figure 3.2: The lower scarp of the Ruinon landslide

3.2 Data

One of the most important preparation aspects of this work was the choice, research, download and organization of the data to be used in the analyses.

In the next sections, the preparation steps of the data are described, and an overview on the final chosen data is provided.

3.2.1 Landslide susceptibility mapping

3.2.1.1 Data preparation

The first step in order to carry out the landslide susceptibility analysis of this thesis project has been to prepare the data that will be used in the analysis.

Data preparation consists of:

- Data collection, i.e. the retrieval of the data, mainly in the form of download from the web;
- Data exploration, i.e. the visualization of the data through a graphical interface;
- Data preprocessing, i.e. the process of modification of the data in order to better suit the needs of the project.

These operations need to be executed for both the areas of study. In this work, the last two tasks were accomplished using QGIS Desktop, a free and open-source geographic information system application that allows users to visualize, modify and analyse geospatial data, supporting a vast assortment of data formats.

In general terms, spatial data can be represented in the form of either vector or raster data. The vector option represents the data employing georeferenced points, lines and polygons; on the other hand rasters are composed of pixels arranged in a grid-type architecture. Therefore, a vector form of representation is convenient when working with discrete spatial features (e.g. borders, roads, rivers, municipalities), while for space-continuous data (e.g. elevation, temperature, but also aerial photographs and satellite images) the raster format is preferred.

Figure 3.3 and Figure 3.4 show the QGIS maps of the two AOIs, highlighting some of the vector layers.

Taking advantage of QGIS powerful editing tools, the series of processes listed below was applied to all the layers, preparing and adjusting them to be later employed in the analysis.

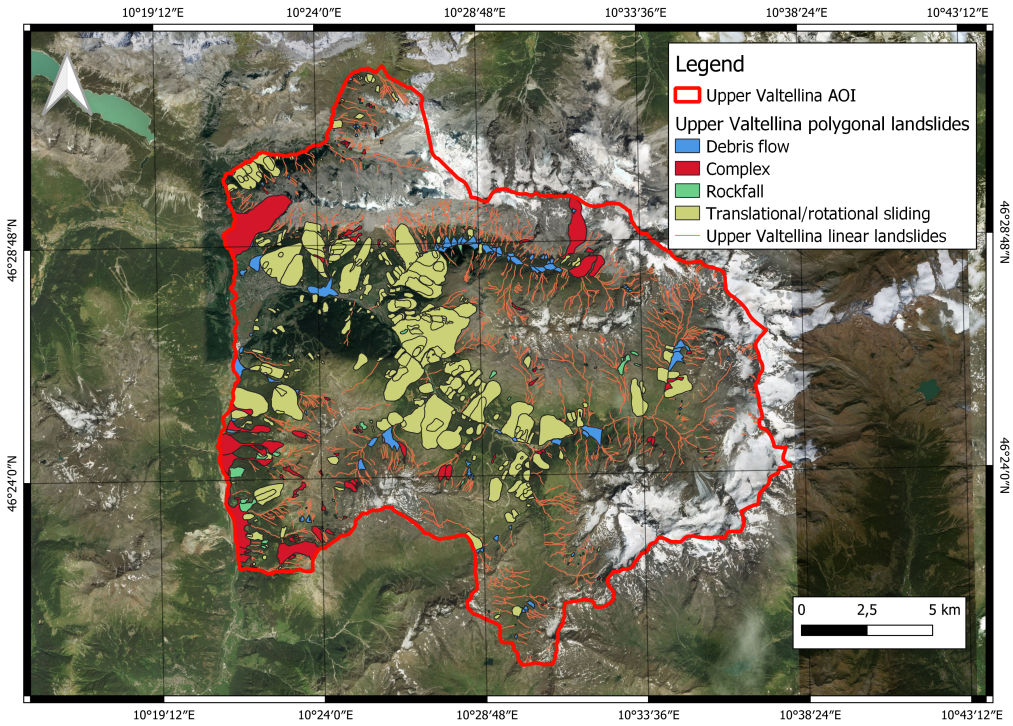


Figure 3.3: Some of the vector data in the Upper Valtellina area

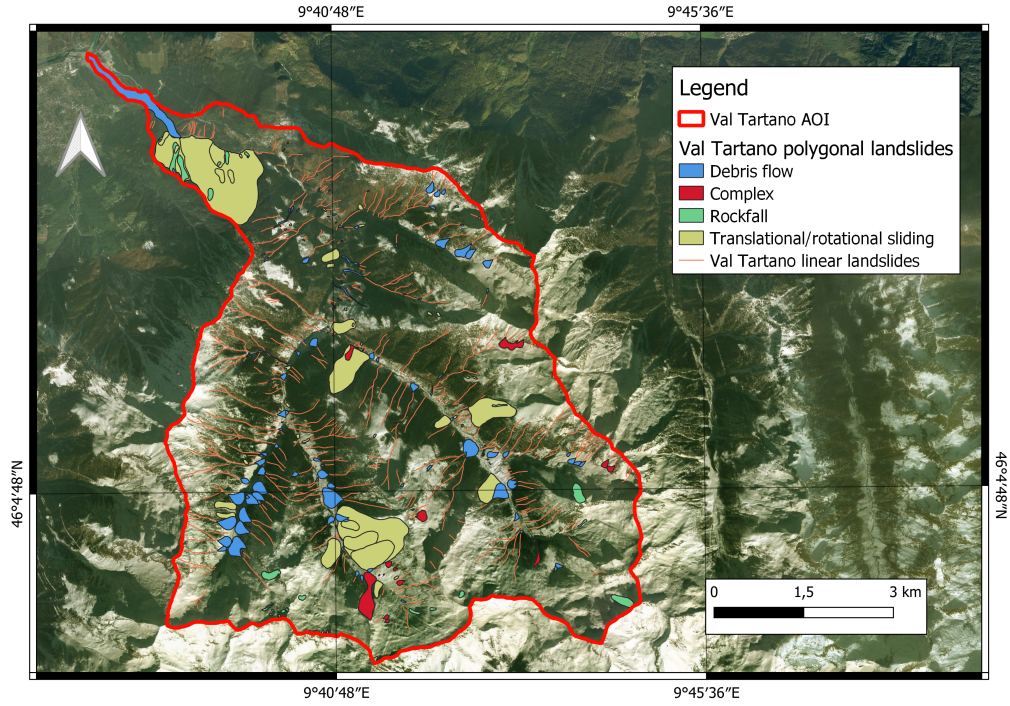


Figure 3.4: Some of the vector data in the Val Tartano area

Reprojection

A Coordinate Reference System (CRS) is a system *"for uniquely referencing spatial information in space as a set of coordinates [...] based on a geodetic horizontal and vertical datum"* (Infrastructure for Spatial Information in Europe, 2019). In order to maintain consistency amongst the dataset, it is necessary to make sure that all layers share the same CRS. In particular, the coordinate system selected for the project is *WGS 84 / UTM zone 32N (EPSG:32632)*, the official reference system for Northern Italy. Conveniently, QGIS contains a feature called *on the fly reprojection*, which allows layers that are added to a project to be automatically projected in the CRS of choice. Nonetheless, the CRS of each layer was checked in order to make the whole dataset consistent.

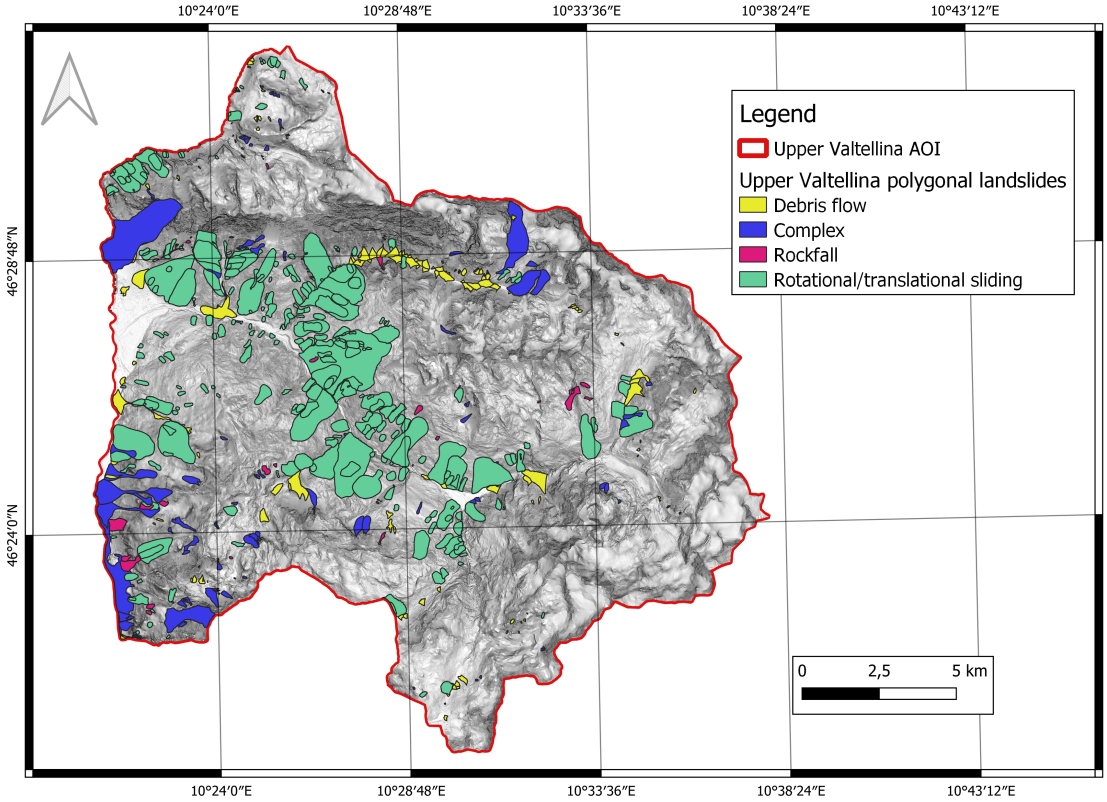
Clipping

Almost every layer downloaded covers either the whole Lombardy region or the Sondrio province, which is the province containing both areas of interest. The downloaded layers have therefore a much larger extent than the study territory; directly using these data would make the computation much slower, while also increasing the memory space occupation of the features. To overcome these problems, the *clipping* operation can be applied to all the layers, both vector and raster ones. This process isolates all the features of a dataset that fall within the polygons of a certain overlay layer. Once clipped using the extent of the Tartano basin and of Upper Valtellina as boundaries, the layers contain only the necessary features for the project.

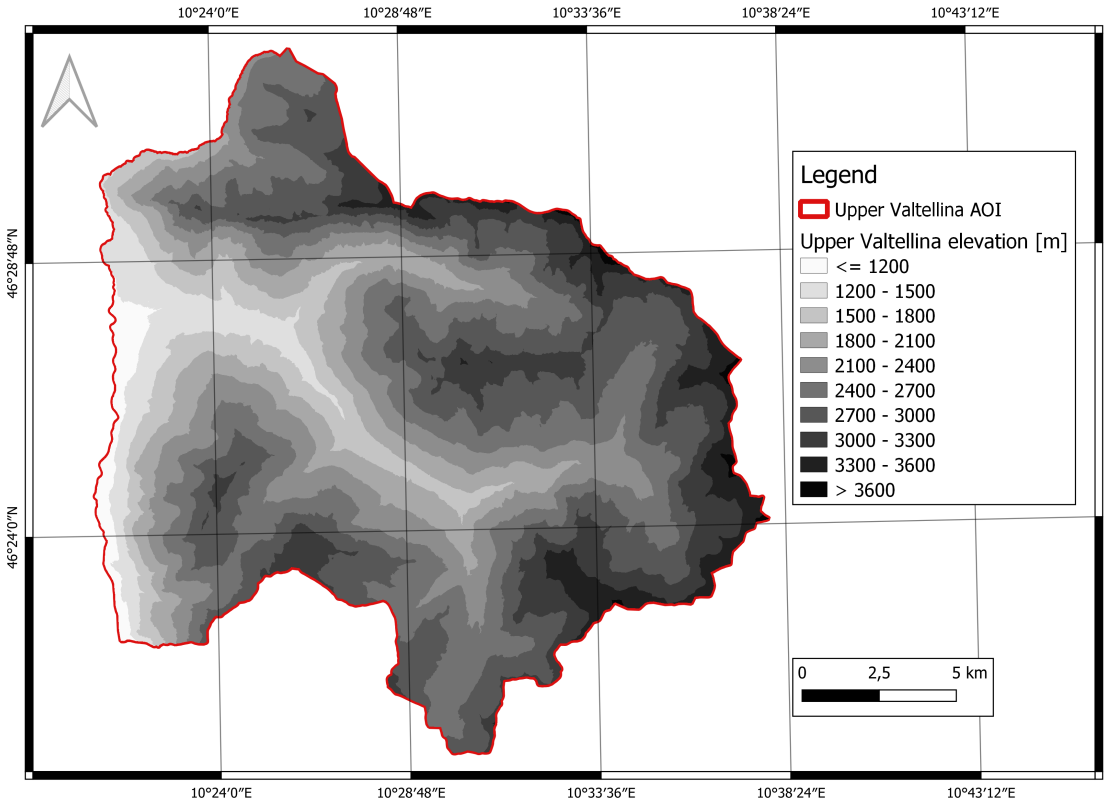
Styling

The visual style of layers can be fully customized using QGIS, in order to aid the understanding of the features described by each layer, and make maps more clear, understandable and readable. In the case of this study, the styling tools offered by QGIS were used mainly for:

- categorizing vector data based on one of their attributes (Figure 3.5a);
- styling vector data to be humanly associable with the real life features they represent (e.g. the river network's colour was chosen to be blue);
- classifying raster data using a discrete interval of values (Figure 3.5b).



(a) Example of vector categorization



(b) Example of raster classification

Figure 3.5: Styling examples

3.2.1.2 Data overview

The data required to carry on the analysis is entirely free and open source, available on the web and distributed by *Geoportale della Regione Lombardia* under IODL 2.0 licence[1], *Istituto Superiore per la Protezione e la Ricerca Ambientale IdroGEO* under CC BY-SA 4.0 license[2] and *Agenzia Regionale per la Protezione Ambientale ARPA Lombardia* under CC BY 4.0 license[3].

Vector data Vector data (Table 3.1) are used to represent discrete spatial features, using the shapefile format, an Esri data format that allows *"[...] storing location, shape and attributes of geographic features"* (ArcGIS documentation, 2021).

Data	Type	Source
General land use	Polygon	GeoPortale Lombardia
Area of Interest	Polygon	Custom made
Fault lines	Line	GeoPortale Lombardia
Geology	Polygon	GeoPortale Lombardia
Inventory of landslides	Multiple Type	IFFI
Streams wet area	Polygon	GeoPortale Lombardia
Road elements	Polygon	GeoPortale Lombardia

Table 3.1: Vector data

Area of Interest
This layer contains the boundaries of the area of interest. This is also the layer used as overlay when clipping the other datasets.

Road elements
This layer shows the road network.

[1] *Italian Open Data License*. It allows the user to use, modify and share the data with the obligation to cite the original source

[2] *Attribution-ShareAlike 4.0 International CC BY-SA*. It allows the user to share and adapt the data with the obligation of attributing and sharing-alike any work

[3] *Attribution-ShareAlike 4.0 International CC BY*. It allows the user to share and adapt the data with the obligation of attributing any work

Streams wet area (River network)
This layer includes all areas which contain water and are part of stream beds.

Geology
This layer contains the lithology classification of the terrain.

Fault lines
This layer contains the fault lines.

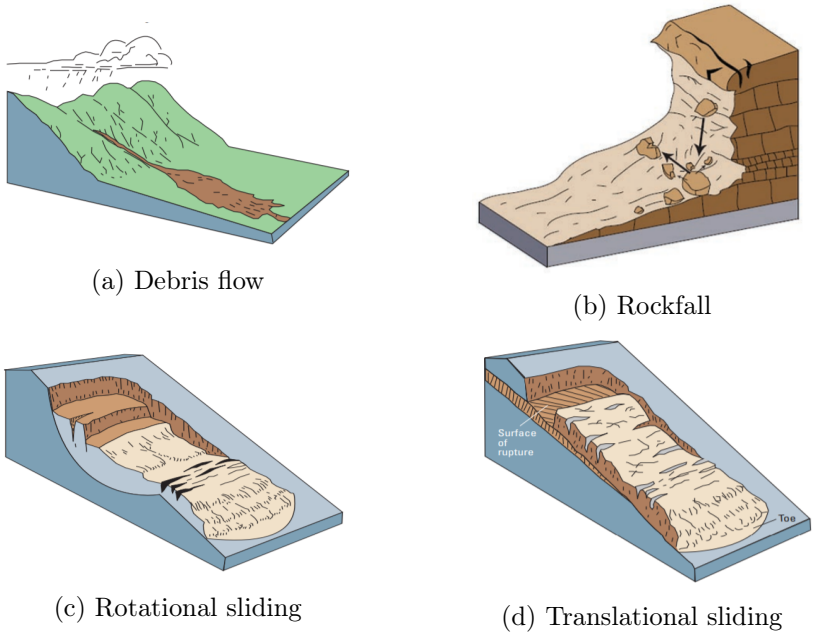
General land use
This layer contains a classification of the territory with respect to the main categories of land use.

Inventory of landslides
The landslide inventory provided by IFFI is an inventory at the scale 1:10000 and updated to the year 2017. It is composed of five different layers, each one depicting in a particular way the spatial allocation of existing landslide phenomena:

- **Identification Point of Landslide Phenomenon**: point layer in which each point is placed in correspondence of the crown of a landslide;
- **Polygonal landslides**: layer containing landslides that have a large enough surface in order to be mapped as polygons. This layer also highlights the type of movement of the landslide: according to the most common landslide classification proposed by Cruden and Varnes (1996), these types can be divided in:
  - Debris flow (Figure 3.6a): typically characterized by soil or fragmented rock moving in a spatially continuous way; this movement is mainly due to water presence in the debris. Debris flow can reach very high velocities when the material loses cohesion, gains water or when the terrain is steep;
  - Rockfall (Figure 3.6b): a very rapid movement where detached material (rock or soil) falls mainly through air because of Earth's gravitational attraction;
  - Rotational/translational slide (Figure 3.6c and Figure 3.6d): downslope movement of rock or soil that is typically occurring along surfaces of

rupture. The difference between rotational and translational slide lies in the form of the surface of rupture: for the rotational kind the surface is curved and concave, for the translational is planar or ondulating;

– Complex: landslides that contain more than one type of movement.



(a) Debris flow (b) Rockfall

(c) Rotational sliding (d) Translational sliding

Figure 3.6: Different types of landslides (Highland and Bobrowsky, 2008)

- **Linear landslides**: layer containing landslides that are not large enough to be mapped as an area and are therefore represented as lines. In the areas of interest the entirety of this layer is composed by debris flow landslides that take place in narrow valleys and channels;
- **Widespread landslide areas**: layer consisting of areas that contain numerous landslide phenomena;
- **Deep-seated Gravitational Slope Deformations**: portions of the territory that are characterized by DGSDs.

Raster data A raster consists in a grid of equally sized pixels, each one containing a value representing information. Rasters can be digital aerial photographs, imagery from satellites, digital pictures, or even scanned maps. Raster data utilized in this thesis are summarized in Table 3.2.

Data	Resolution [m]	Source
DTM	5x5	GeoPortale Lombardia
Slope	5x5	Computed
Aspect	5x5	Computed
Eastness	5x5	Computed
Northness	5x5	Computed
Profile curvature	5x5	Computed
Plan curvature	5x5	Computed
NDVI	5x5	Computed
Precipitation	1500x1500	ARPA Lombardia
TWI	5x5	Computed

Table 3.2: Raster data

Digital Terrain Model (DTM)

This is the 2015 Digital Terrain Model for the Lombardy region; each pixel value therefore represents the elevation of that pixel. Starting from the DTM, other rasters were calculated:

- **Slope:** a map that represents the maximum rate of elevation change between each cell of the DTM;
- **Aspect (Figure 3.7):** a representation that identifies the down-slope direction that each of the cell faces;

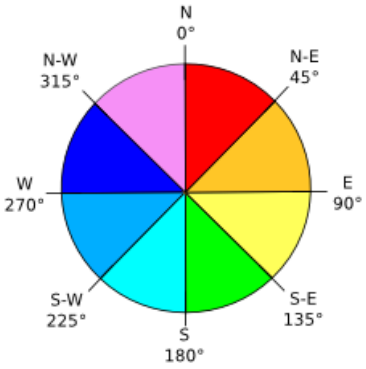


Figure 3.7: Aspect diagram

- **Eastness:** obtained by applying a *sin()* function to the Aspect, Eastness ranges from -1 to 1 and describes whether a slope faces the East direction (Eastness = 1) or the West direction (Eastness = -1);
- **Northness:** obtained by applying a *cos()* function to the Aspect, Northness ranges from -1 to 1 and describes whether a slope faces the North direction (Northness = 1) or the South direction (Northness = -1);
- **Profile curvature (Figure 3.8):** a negative value of this parameter indicates that the surface is upwardly convex at that cell, a positive value indicates that the surface is upwardly concave at that cell, and a value of zero indicates that the surface is linear;

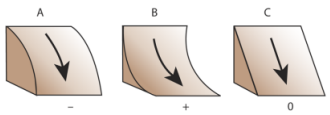


Figure 3.8: Profile curvature

- **Plan curvature (Figure 3.9):** a negative value of this parameter indicates that the surface is laterally concave at that cell, a positive value indicates that the surface is laterally convex at that cell, and a value of zero indicates that the surface is linear;

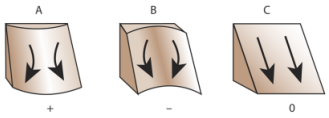


Figure 3.9: Plan curvature

- **Topographic Wetness Index (TWI):** hydrological index that describes the tendency of an area to accumulate water. It is computed using the formula

$$TWI = ln(\frac{\alpha}{tan(\beta)}) \tag{3.1}$$

  where α is the Specific Catchment Area and $tan(\beta)$ is the slope.

Normalized Difference Vegetation Index (NDVI)

The NDVI is an index useful to determine whether an area contains vegetation or not. It can be calculated from satellite images, using the formula

$$NDVI = \frac{NIR - Red}{NIR + Red} \tag{3.2}$$

where *NIR* is the spectral reflectance measured in the Near InfraRed region, and *Red* is the spectral reflectance measured in the visible red region. In this study, the NDVI has been computed using Google Earth Engine©. The code followed this procedure:

- Satellite images from the Sentinel-2 mission overlapping the area of interest were accessed for the whole year of 2020. In this process, a cloud filter was also applied;
- For each month, the NDVI formula was applied with the mean values of the *NIR* and *Red* bands;
- The 12 resulting NDVI maps were merged in order to obtain a yearly mean NDVI map;
- The final map was exported to Google Drive with the correct resolution.

In B.1.1 you can find the GEE code written for the computation and export of the NDVI map.

Precipitation

This raster was obtained starting from data collected from an interpolation of hourly observations coming from the regional meteorological network of ARPA for the region of Lombardy. For each hour, a `.txt` file containing the values from the observations in the meteorological station interpolated on a 1.5x1.5 km grid was produced. Using a custom Python script, all the data for the year 2020 were averaged and then converted to a `.csv` file, thus obtaining the 2020 average measure of precipitation for the whole Lombardy region. The `.csv` file was later imported as a raster in QGIS.

In B.1.2 you can find the Python code used for the creation of this raster layer.

3.2.2 Landslide displacement monitoring

For monitoring the displacements of the Ruinon landslide through time, the data required are solely Sentinel-2 satellite images.

3.2.2.1 Data exploration and download

In order to obtain quality results in the displacement analysis, it is important that the satellite images do not contain clouds over the Ruinon landslide, and that the terrain is free from snow. The Sentinel Hub EO browser was used to inspect images over the chosen AOI and select the ones to be downloaded.

The download was made using the official portal of the Copernicus programme, the Copernicus Open Access Hub. For this analysis, Sentinel-2 Level-1C products were considered.

The downloaded images were employed for monitoring the displacements of the considered landslide on a yearly basis (6.2.1, one image per year from 2015 to 2020) and with a focus on the summer months of 2019 (6.2.2). Some of them were instead used for validating the proposed technique.

3.2.2.2 Data overview

Table 3.3 summarizes all the different multiband images downloaded for the displacement monitoring, while Figure 3.10 contains an example of a downloaded Sentinel-2 tile.

Mission	**Sensing date**	**Relative Orbit**	**Tile Number**
Sentinel-2A	03 August 2015	22	T32TPS
Sentinel-2A	27 August 2016	22	T32TPS
Sentinel-2A	09 July 2017	22	T32TPS
Sentinel-2A	08 July 2018	22	T32TPS
Sentinel-2A	16 July 2019	65	T32TPS
Sentinel-2B	18 July 2019	22	T32TPS
Sentinel-2A	23 July 2019	22	T32TPS
Sentinel-2B	27 August 2019	22	T32TPS
Sentinel-2A	11 September 2019	22	T32TPS
Sentinel-2B	16 September 2019	22	T32TPS
Sentinel-2A	21 September 2019	22	T32TPS
Sentinel-2A	01 October 2019	22	T32TPS
Sentinel-2A	18 May 2020	22	T32TPS
Sentinel-2A	26 August 2020	22	T32TPS
Sentinel-2A	15 September 2020	22	T32TPS

Table 3.3: Downloaded Sentinel-2 images

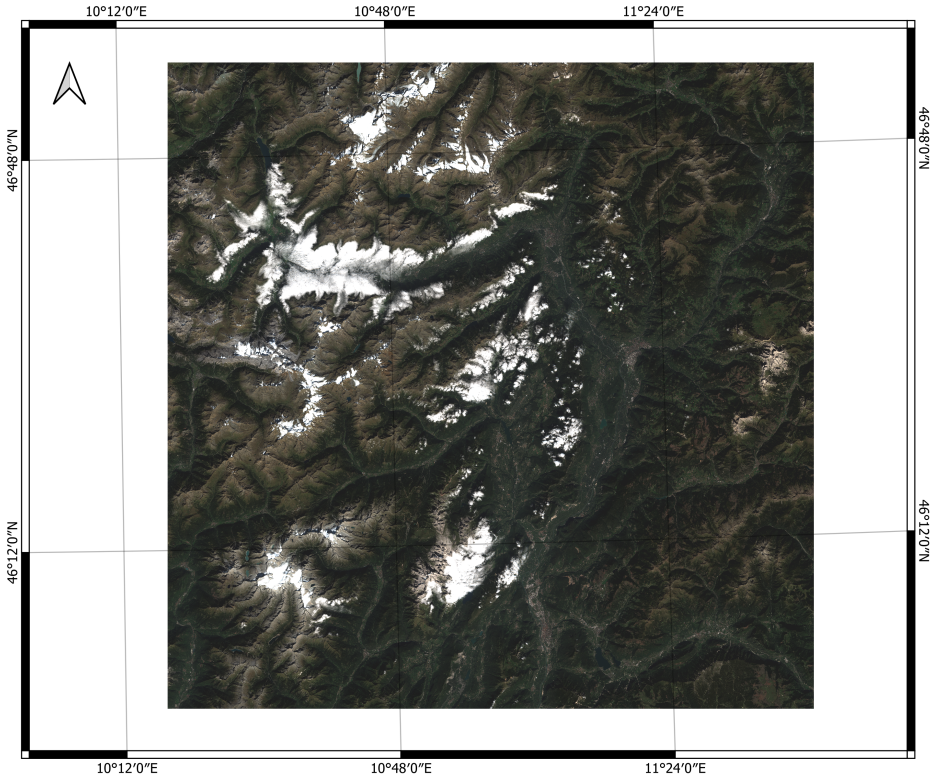


Figure 3.10: Example of one of the downloaded Sentinel-2 Level-1C tiles, in True Colours visualization

Chapter 4

Tools and technologies

Several instruments and techniques were employed for the making of the analyses performed in this work.

4.1 Landslide susceptibility mapping

4.1.1 Software tools

Access to good quality multispectral images is the first requirement for analyses of remote sensing images like the one described in this work. Aside from that, another fundamental aspect is the availability of robust and reliable software tools, capable of consistently processing the aforementioned images. Nowadays, the majority of these tools are available as free and open-source software (FOSS), thus contributing to make the software:

- easy to distribute;
- highly customizable to specific studies' needs;
- more secure and transparent.

Moreover, free and open-source software for geospatial applications (FOSS4G) provide tools and functionalities capable of not only competing but also outclassing their proprietary counterpart (Brovelli et al., 2017).

Table 4.1 illustrates the purposes of each of the software tools subsequently discussed in this section.

Software	Purpose	License
Copernicus Hub	Data exploration and download	FOSS
QGIS	Data visualization and preprocessing	FOSS
Google Earth Engine©	Data preprocessing	Free for research (not open-source)
RStudio	Data processing	Used in its FOSS version
GRASS GIS	Data processing and visualization	FOSS
R	Programming language	//
Python	Programming language	//
JavaScript	Programming language	//

Table 4.1: Software tools and programming languages used for landslide susceptibility analysis

4.1.1.1 QGIS

QGIS is a user-friendly free and open-source Geographic Information System (GIS). It offers numerous functionalities for viewing, editing and analysing geospatial data, along with many additional plugins and the integration with other open-source GIS packages (e.g. GRASS GIS, PostGIS etc.) (QGIS, 2021). In this work, QGIS was used for viewing and exploring data during all the steps of the analysis, due to its powerful viewing/styling tools and easy to use interface, and to carry out some preprocessing operation on the initial data.

4.1.1.2 Google Earth Engine© (GEE)

GEE is a web-platform for cloud-based processing of remote sensing data on large scales (Google, 2021). The combination of its multi-petabyte catalogue of satellite imagery and geospatial datasets with the computational power of Google's servers makes it a very powerful instrument when it comes to processing large images. In particular, in this study it was used for the computation of the NDVI raster map (Normalized Difference Vegetation Index (NDVI)).

4.1.1.3 RStudio

RStudio is an integrated development environment for R and Python, with a console, an editor that supports direct code execution, and tools for plotting, history, debugging and workspace management (RStudio, 2021). The Landslide Susceptibility Analysis with the Random Forests machine learning method was accomplished using RStudio.

4.1.1.4 Programming languages

The analyses described in this project were carried out employing custom scripts; the programming languages of choice are:

- R: a language for statistical computing and graphics, used for the landslide susceptibility analysis with the Random Forests machine learning method (R foundation, 2021);
- Python: the programming language that best interacts with geospatial applications such as QGIS and GRASS GIS; for LSM it was used with a custom script to compute the precipitation layer (Python software foundation, 2021);
- JavaScript: language known for being the scripting language of web-applications; it is also the language used in the Google Earth Engine© environment.

4.1.2 Analysis techniques

4.1.2.1 Random Forests

Random Forests is a machine learning algorithm for prediction and classification; its principle of working is to create multiple decision trees, each one depending from a random vector chosen independently from past random vectors but with the same distribution. After the trees have been created, they all vote for the most popular classification (Breiman, 2001). This was the algorithm of choice for the landslide susceptibility analysis of this work; in practice, the *ModelMap* package (Freeman et al., 2016), implemented in R, was utilized. The *ModelMap* package allows to build predictive models using a Random Forests predictor with the *randomForest* package (Liaw and Wiener, 2002). After having built the model, *ModelMap* also includes the possibility to validate the model and to create maps over large geographic areas. Another fundamental feature is the ability of the package to work simultaneously

with both continuous and discrete variables.

4.2 Landslide displacement monitoring

4.2.1 Software tools

4.2.1.1 Copernicus Open Access Hub

The Copernicus Hub is the official online data portal of the Copernicus programme of the European Commission. It grants access to all Sentinel missions data through an interactive Graphical User Interface, including the possibility of:

- searching through the data using a text bar;
- filtering the data according to various parameters (mission, sensing period, ingestion period, orbit number etc.);
- definition of an AOI in order to filter the results to only those that overlap the defined area;
- exploring the data and their additional information;
- downloading satellite images. (Copernicus Programme, 2021)

In this work, Copernicus Hub was utilized to find and download Sentinel-2 images relevant to the case study.

4.2.1.2 Sentinel Hub EO browser

The EO browser functionality of the Sentinel Hub allows to browse, visualize and compare full resolution images from all the Sentinel satellite collections. It is possible to specify a window of interest, a time range, a maximum cloud coverage and inspect the resulting data in the browser. Another feature that was used is the possibility of setting different band combinations, both pre-made or custom ones, for the visualization of the images.

4.2.1.3 GRASS GIS

Geographic Resources Analysis Support System, commonly referred to as GRASS GIS, *"[...] is a Geographic Information System (GIS) technology built for vector and*

raster geospatial data management, geoprocessing, spatial modelling and visualization" (OSGeo, 2021). In this work GRASS GIS was mainly used for the execution of custom Python scripts that included GRASS functions.

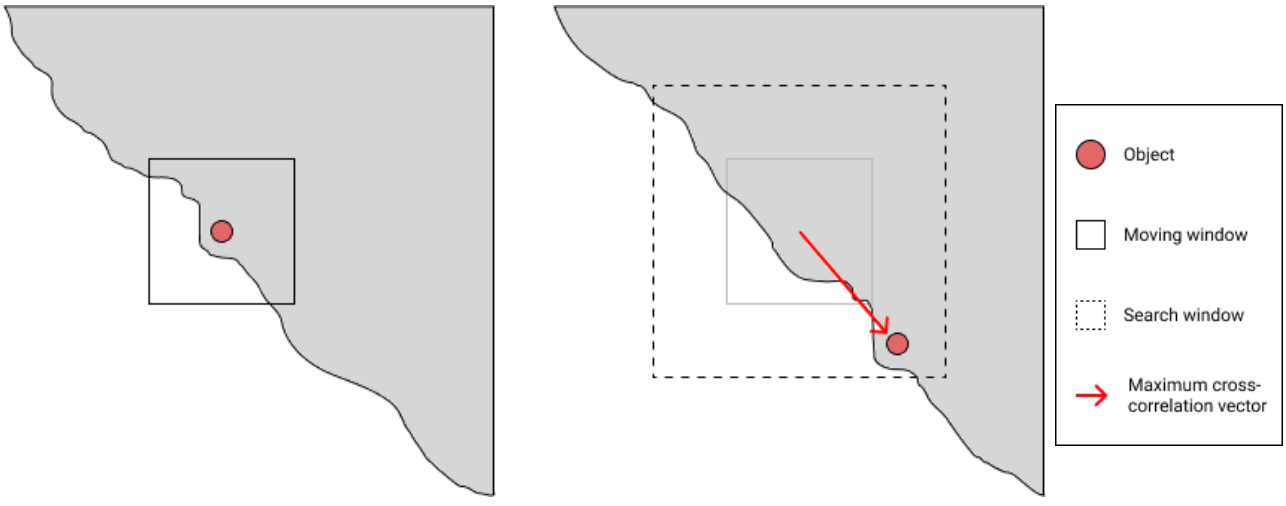
4.2.1.4 Python

The powerful integration between Python and GRASS GIS was exploited for developing the procedure for the Ruinon landslide displacement analysis.

4.2.2 Analysis techniques

4.2.2.1 Maximum Cross-Correlation

The Maximum Cross-Correlation method is used in this work to detect moving pixels between two temporal images based on a cross-correlation coefficient. The process starts by placing a square window, also referred to as *template*, on the same position of the two images. Then, the cross-correlation coefficient is computed for all possible windows obtained by shifting the original template on the *slave* image in all directions. This shifting process is limited to the extent of the original window. At this point, the shifted window which has the highest cross-correlation coefficient with the template on the *master* image is selected (Figure 4.1): a displacement vector between the centres of these two windows can be computed (You et al., 2017).



(a) *Master* image (b) *Slave* image

Figure 4.1: Maximum Cross-Correlation procedure example

The cross-correlation coefficient can be computed using the following formula:

$$\rho(i,j) = \frac{cov[A(x_m, y_m), B(x_s, y_s)]}{\sqrt{var[A(x_m, y_m)var[B(x_s, y_s)]])}} \tag{4.1}$$

where the subscript m indicates the *master* image and the subscript s the *slave image*. Therefore, $A(x_m, y_m)$ is the set of pixels of the *master* image inside the window centred in (x_m, y_m), while $B(x_s, y_s)$ is the set of pixels of the *slave* image inside the window centred in (x_s, y_s), or in other terms $(x_m + i, y_m + j)$.

Chapter 5

Methodology

5.1 Landslide Susceptibility Mapping (LSM)

The first aim of this work was to use Machine Learning tools to create maps depicting the landslide susceptibility for the two AOIs. Landslide susceptibility denotes the probability level to be prone to mass movements (Yordanov and Brovelli, 2020a). The underlying concept to landslide susceptibility maps is that areas that have similar environmental conditions with respect to known landslides are more likely to be susceptible to similar phenomena. To carry out this analysis, the Random Forests machine learning algorithm was chosen (Breiman, 2001), since it was proven to outperform other statistical-based methods in Landslide Susceptibility assessment (Yordanov and Brovelli, 2020a).

5.1.1 Data preprocessing

A crucial part of any Machine Learning analysis is the creation and preparation of the dataset that will constitute the foundation for the model. In particular, for Landslide Susceptibility analyses it is necessary to choose a suitable set of environmental variables that are considered to be related to the occurring of mass movements, and it is as well necessary to have a large enough landslide phenomena inventory.

5.1.1.1 Factors selection and preparation

The terrain variables to take into consideration for the analysis were chosen amongst the ones that are, according to literature (Guzzetti et al., 2006; Reichenbach et al., 2018; Zhou et al., 2018; Emami et al., 2020; Fang et al., 2020; Yordanov and Brovelli, 2020a), most likely to cause or accelerate landslide phenomena. Namely, the initial chosen factors were:

- Elevation (Raster data)
- Slope (Raster data)
- Eastness (Raster data)
- Northness (Raster data)
- Distance from roads (Vector data)
- Distance from rivers (Vector data)
- Distance from faults (Vector data)
- Topographic Wetness Index (Raster data)
- NDVI (Raster data)
- Land use (Vector data)
- Lithology (Vector data)
- Plan curvature (Raster data)
- Profile curvature (Raster data)
- Precipitation (Raster data)

These variables need to be in raster format and to have the same pixel size; the first step to satisfy these constraints is to convert vector data to raster. The data that needed conversion were:

- Road network: in order to obtain a raster showing the distance from roads, the road network layer was first buffered at 50 m, 100 m, 200 m, 500 m, and then the remaining zone was considered to have a distance >500 m from the

nearest road. The buffered layer was then converted to a raster using QGIS (Figure 5.1e);

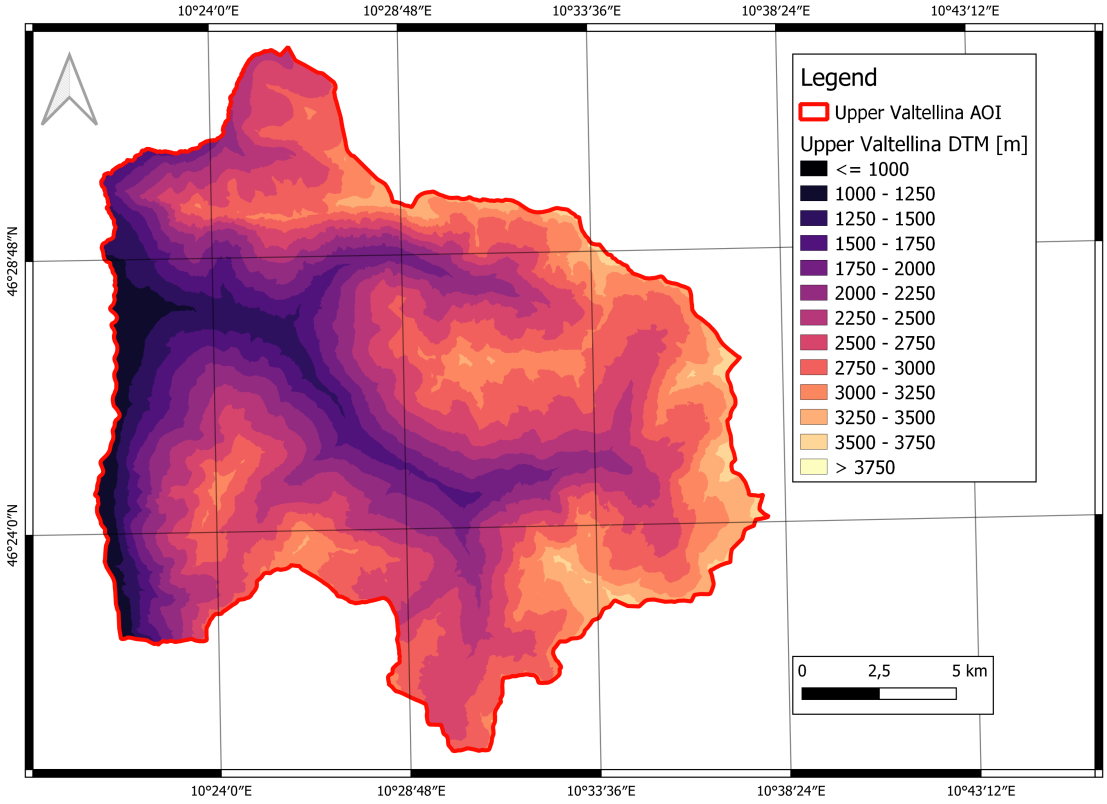
- River network: the same process applied to the road network was used for the river network (Figure 5.1f);
- Fault lines: similarly to the road and river network, the fault lines were first buffered and then converted to raster (Figure 5.1g);
- Land use: the different categories of land use were categorized in a numerical way and the vector layer was then rasterized (Figure 5.1j);
- Lithology: similarly to the land use, the lithology features were first categorized and then the layer was converted to raster (Figure 5.1k);

It is important to note that this analysis took advantage of the capability of the Random Forests algorithm to work with both discrete and continuous variables. Table 5.1 illustrates the categorization employed for discrete factors. Note that the area of Val Tartano has less lithology classes than Upper Valtellina, due to the absence of some rock types, namely: limestone, dolostone, gypsum and anhydrite, filonian rocks, serpentinites and schists, marbles.

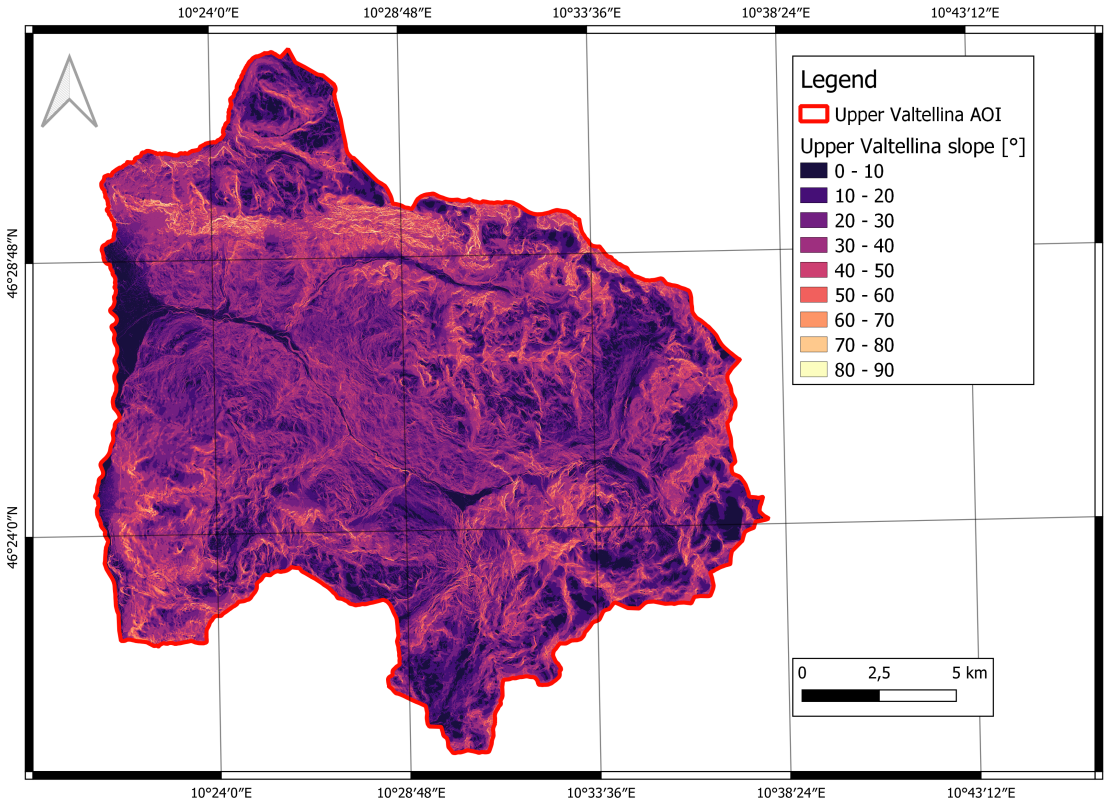
Factor	**Number of classes**	**Classification parameter**
Distance from roads	5	Buffered distance
Distance from rivers	5	Buffered distance
Distance from faults	5	Buffered distance
Land use	11	Level 2 classification of land use
Lithology	14 - Upper Valtellina 8 - Val Tartano	Lithology characterization

Table 5.1: Categorical environmental variables

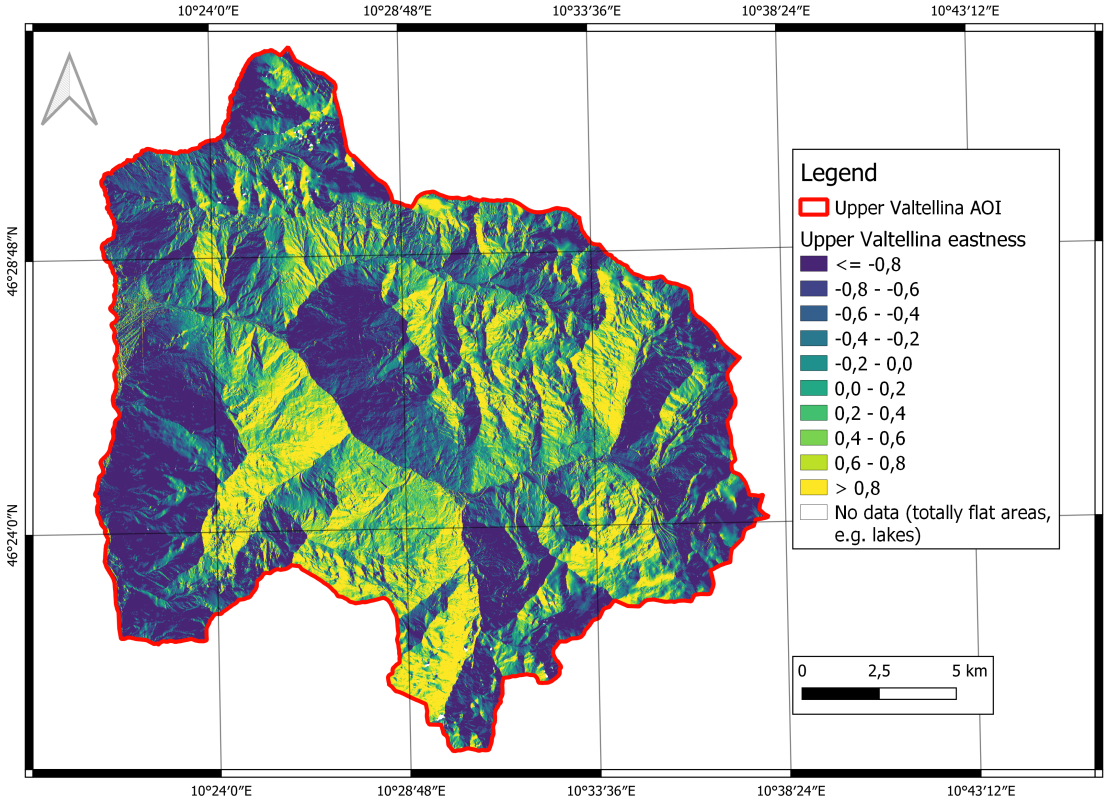
The following pages show the final preprocessed factors, that were used later on in the analysis:



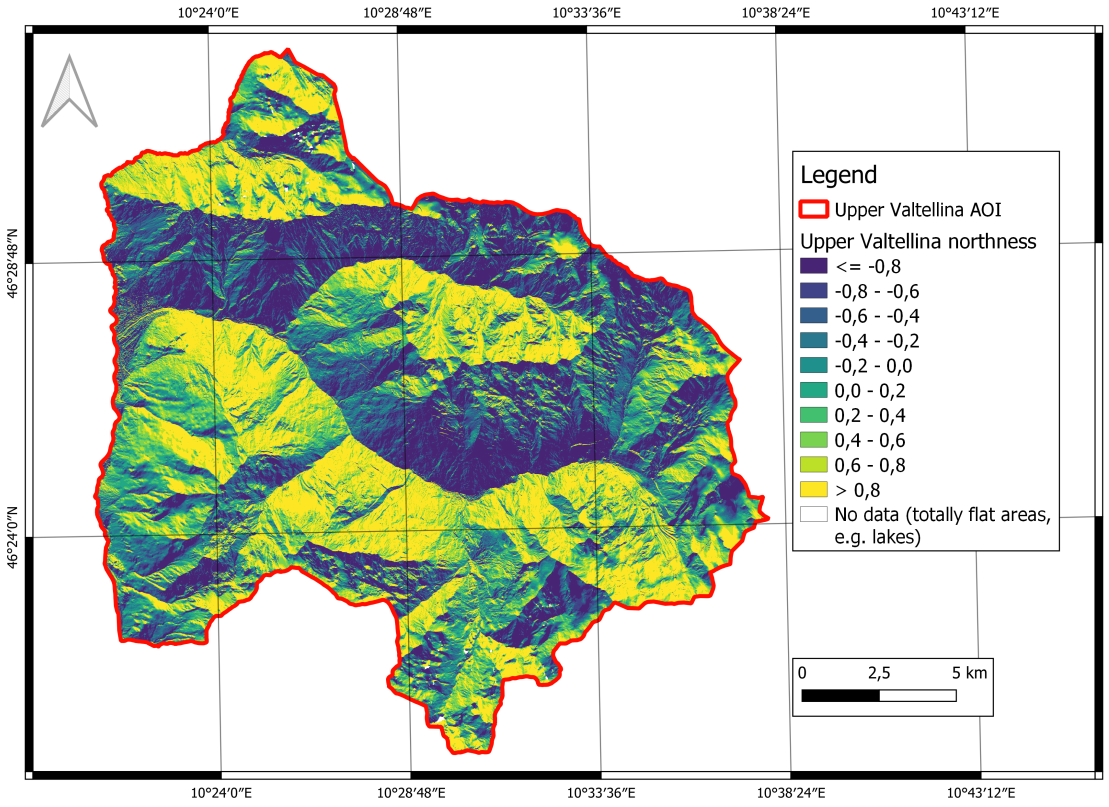
(a) Upper Valtellina DTM



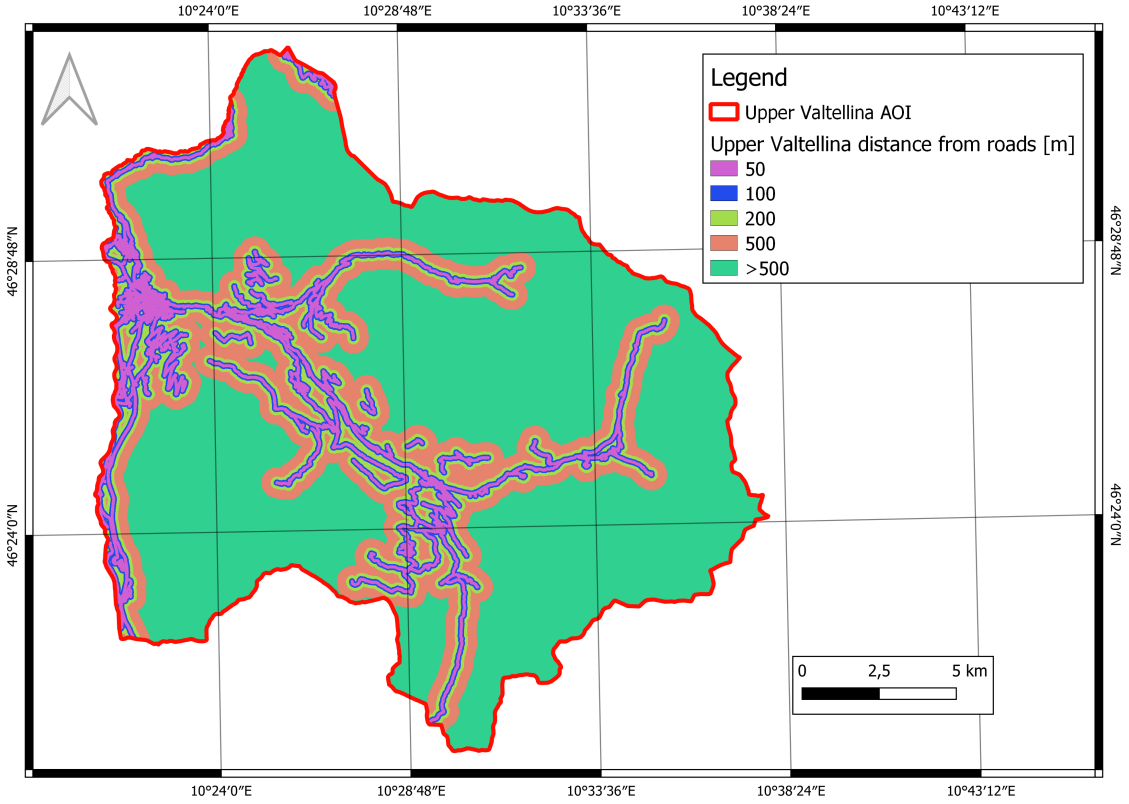
(b) Upper Valtellina slope



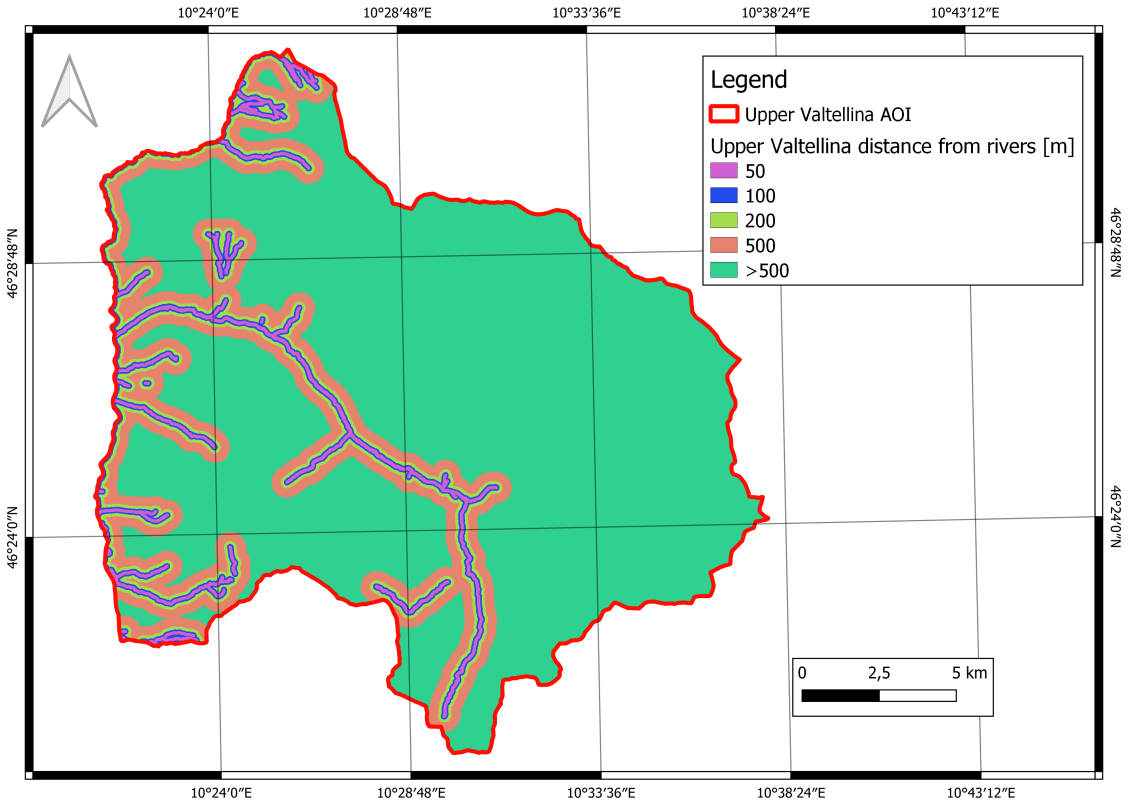
(c) Upper Valtellina eastness



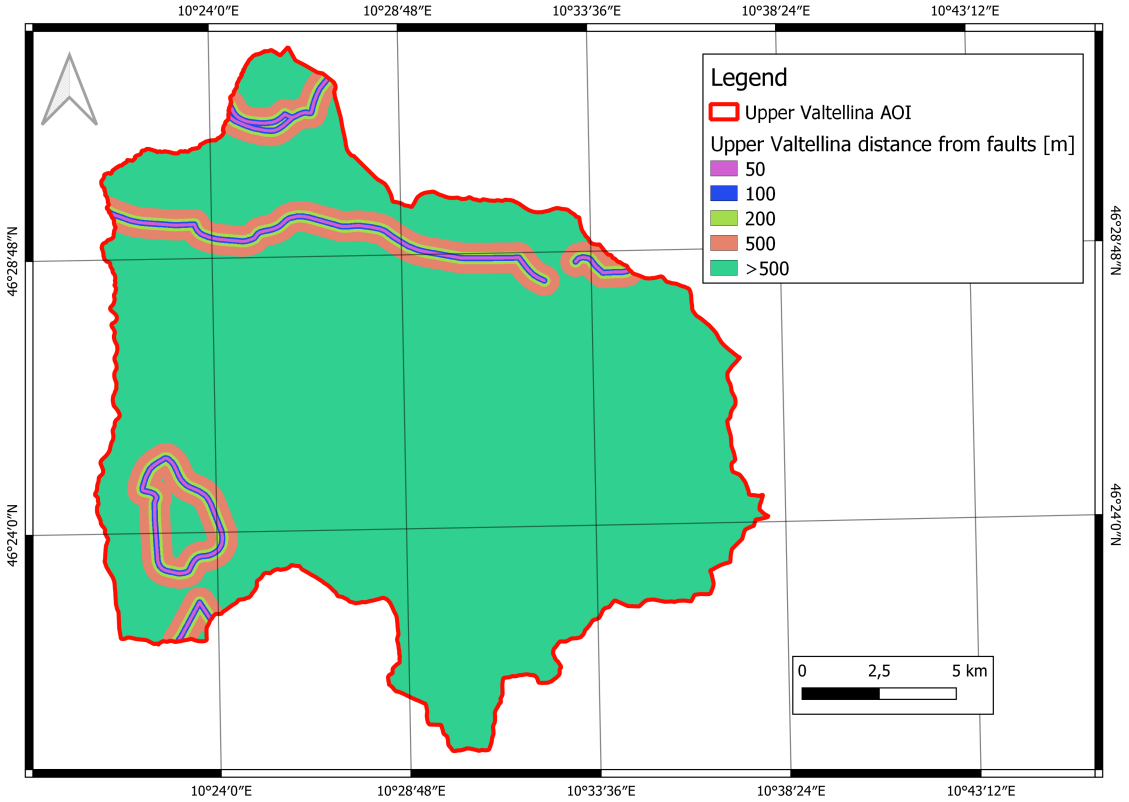
(d) Upper Valtellina northness



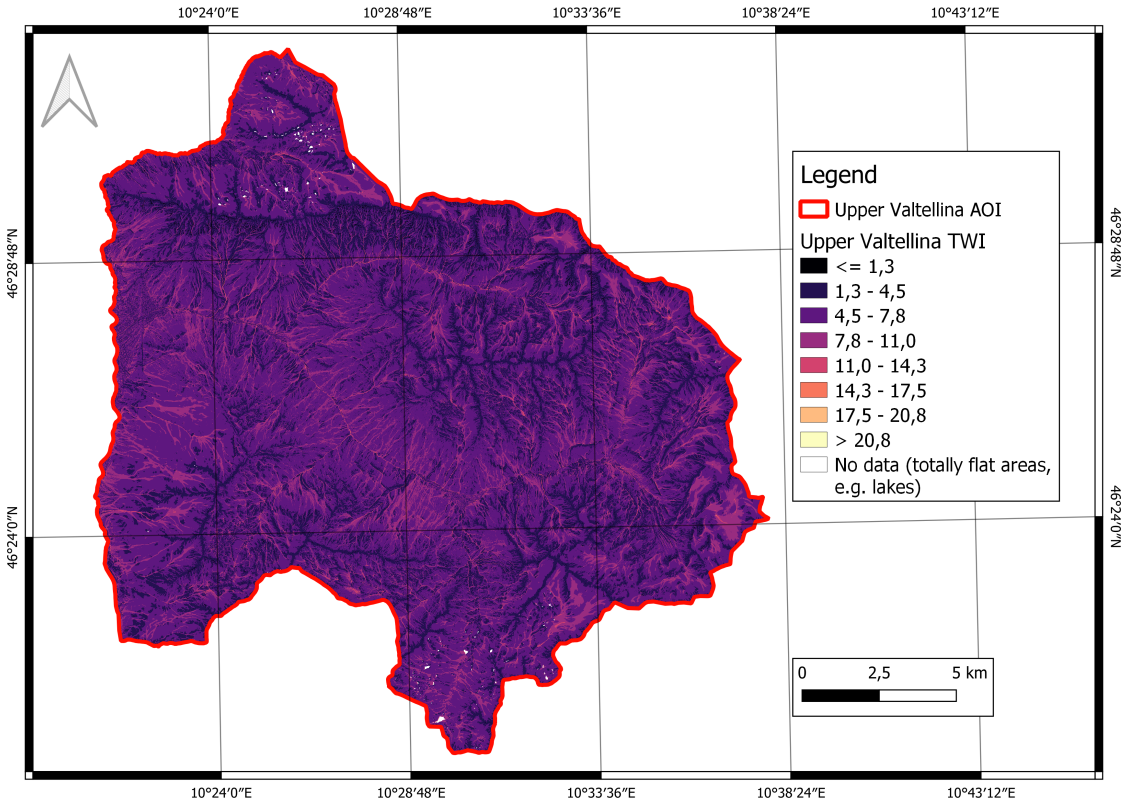
(e) Upper Valtellina distance from roads



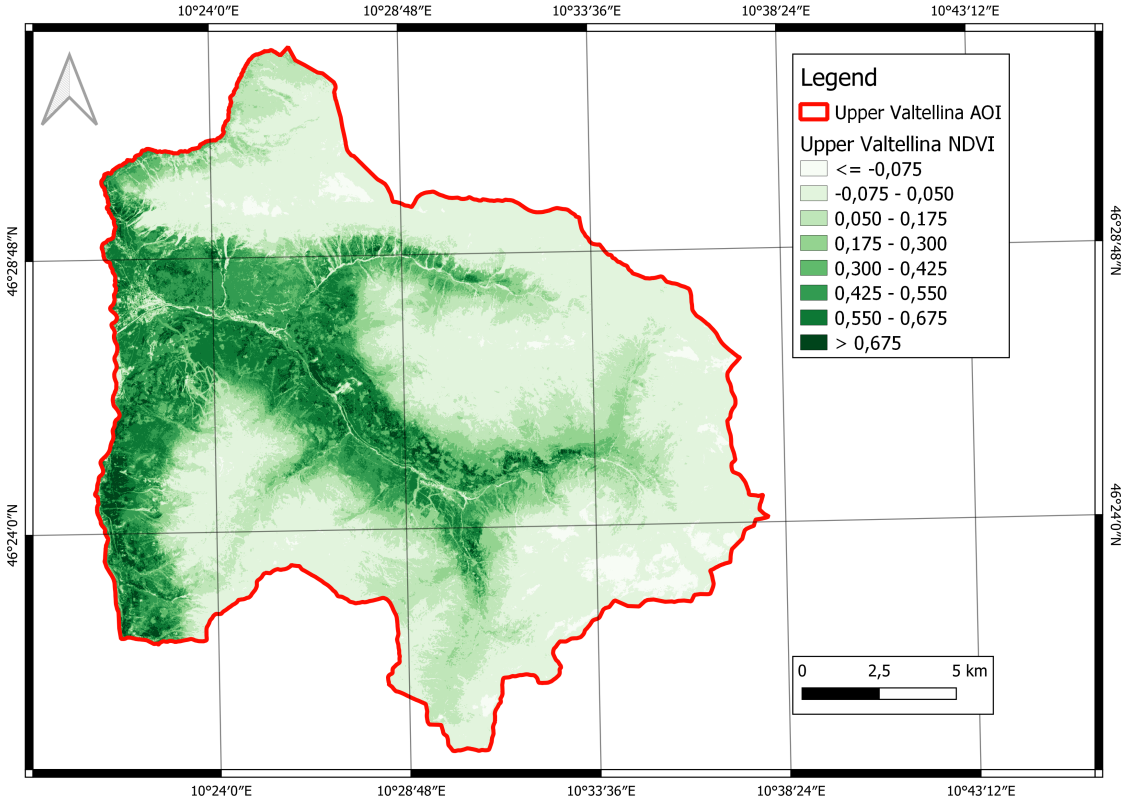
(f) Upper Valtellina distance from rivers



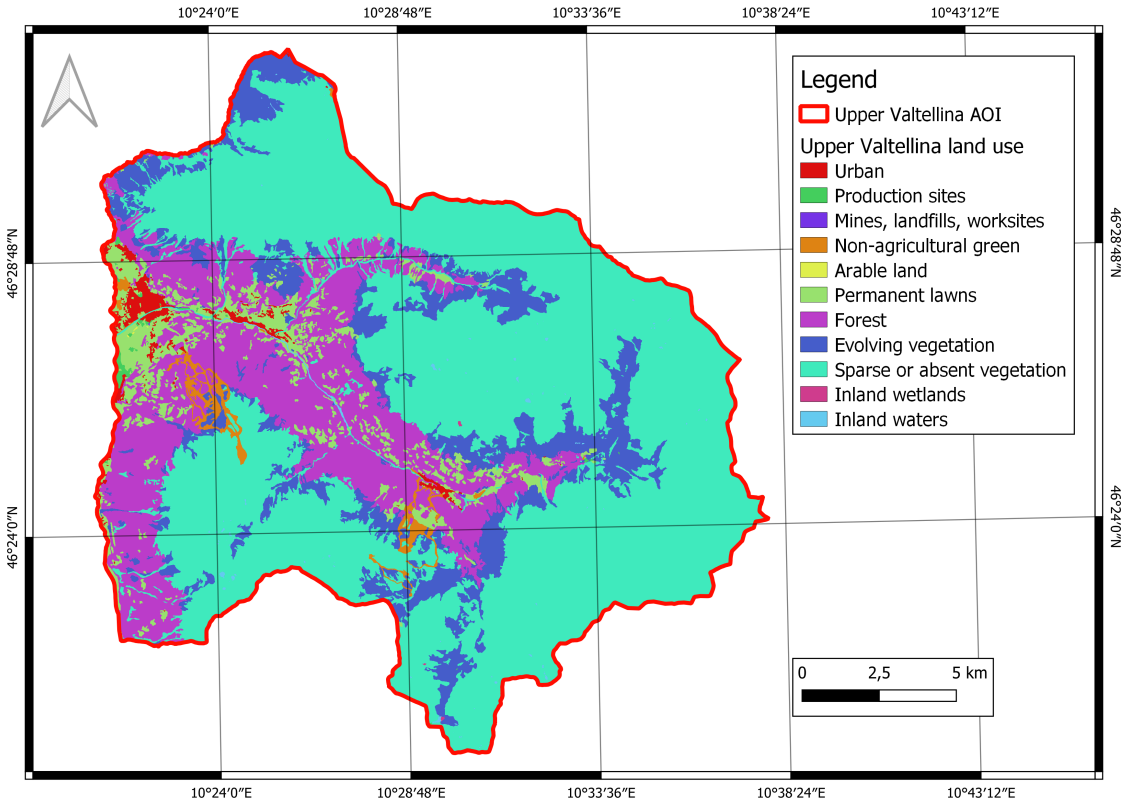
(g) Upper Valtellina distance from faults



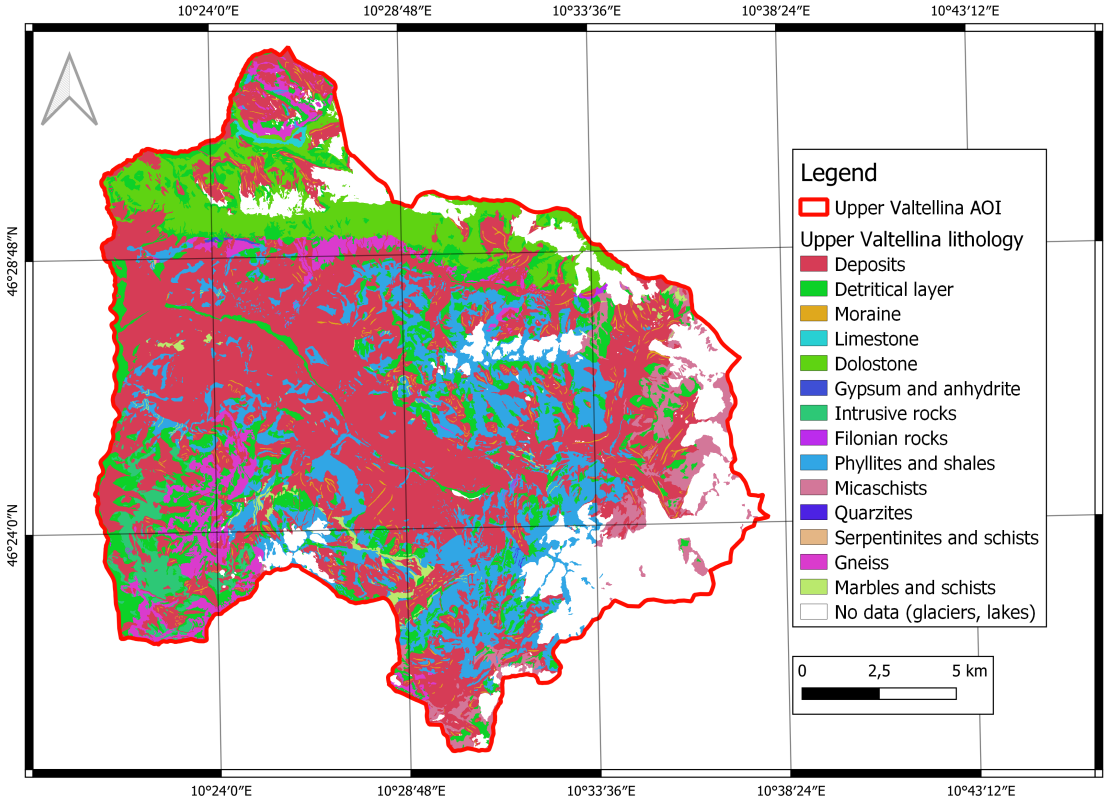
(h) Upper Valtellina Topographic Wetness Index



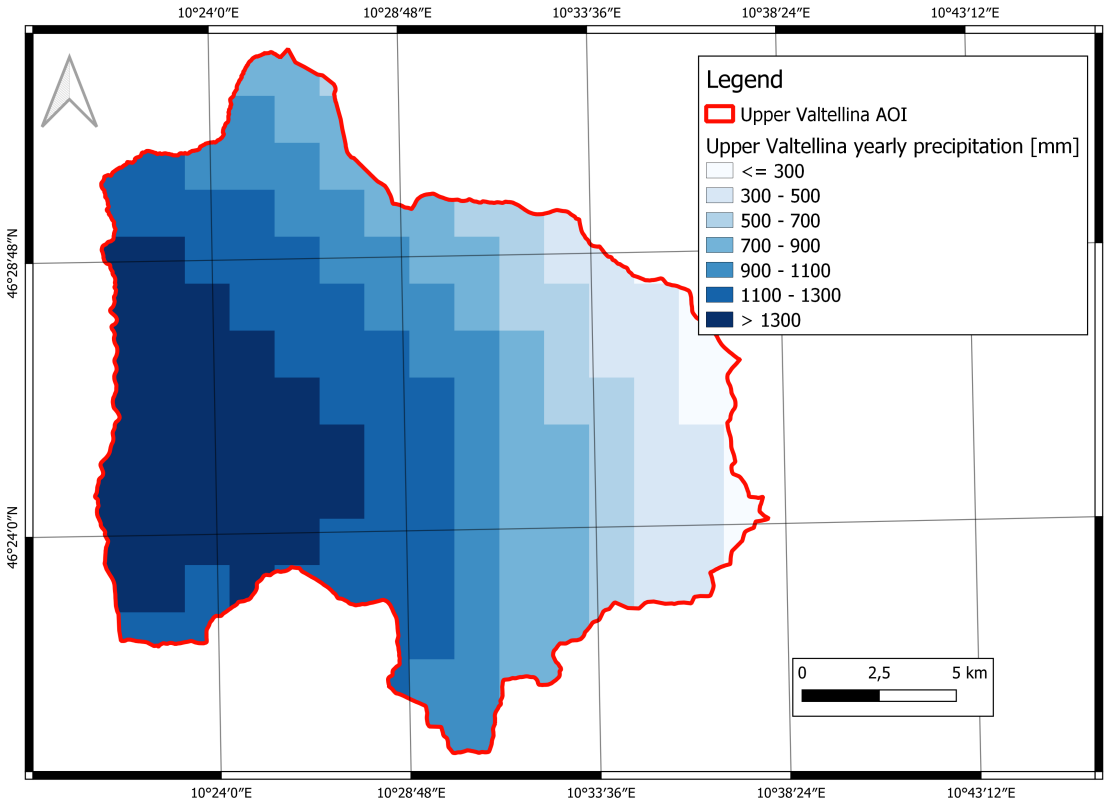
(i) Upper Valtellina NDVI



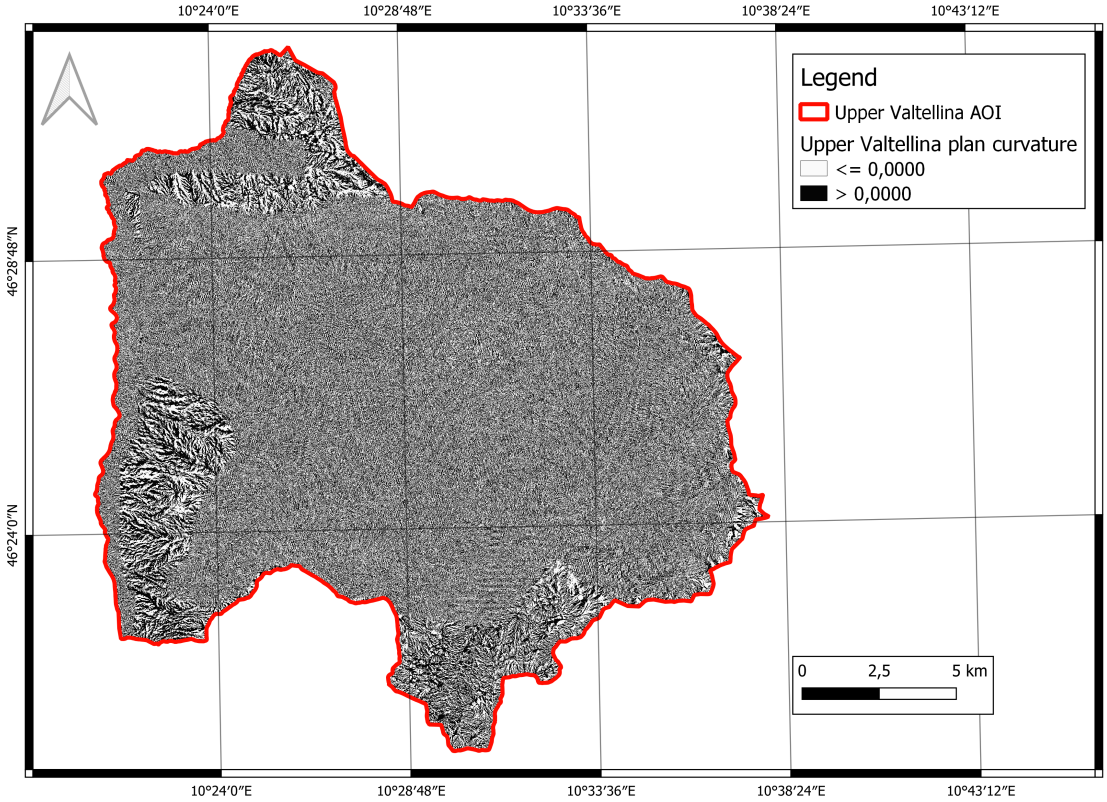
(j) Upper Valtellina land use



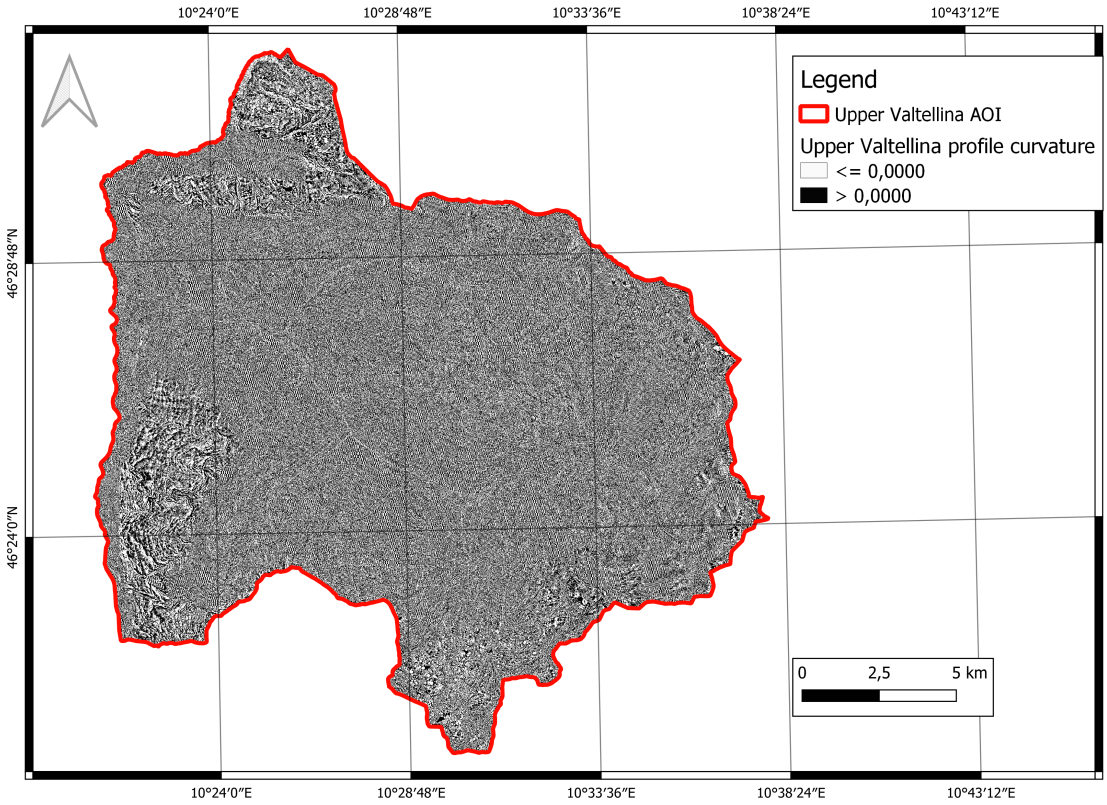
(k) Upper Valtellina lithology



(l) Upper Valtellina yearly precipitation



(m) Upper Valtellina plan curvature



(n) Upper Valtellina profile curvature

Figure 5.1: Upper Valtellina terrain variables

Please refer to Appendix A for the images of the terrain variables of Val Tartano.

After a few initial trials, the list of factors to be considered in the analysis was revised taking into consideration the first outputs. In particular, it was seen that the precipitation map, since it has a coarse grid (the original image has a grid size of 1.5x1.5 km), had a negative effect on the results, yielding a map with abrupt susceptibility changes along the grid of the precipitation data (Figure 5.2). Because of this, precipitation was excluded from the considered factors.

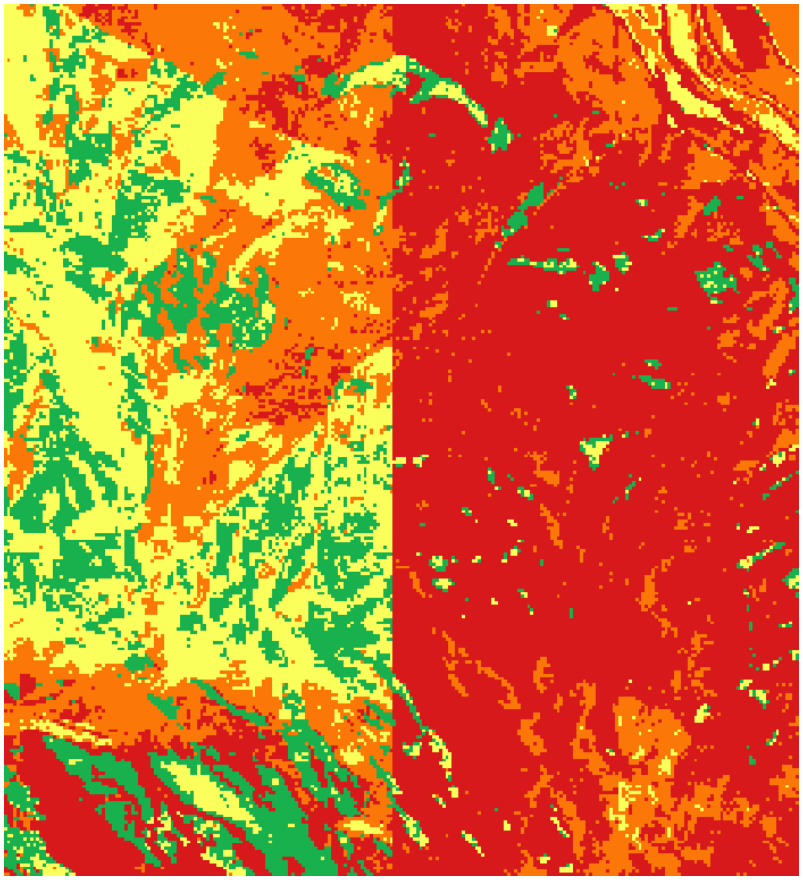


Figure 5.2: The abrupt susceptibility changes produced by the precipitation factor

Mapping unit

The choice of an appropriate mapping unit is a fundamental aspect of landslide susceptibility mapping (Guzzetti et al., 1999; Reichenbach et al., 2018). In fact, once the conversion to raster was completed, the factors rasters were evened in terms of dimensions and pixel size. Then, for all the analyses, the DTM raster

was taken as reference, and the mapping unit was chosen to be the DTM native resolution, i.e. 5x5 m. This approach was adopted in the first place because some terrain variables were already calculated starting from the DTM, but also because this resolution was coherent for the representation of the other variables (such as roads, rivers, faults). Lastly, it was also acceptable for representing the landslide features, both the polygonal and the linear ones.

5.1.1.2 Landslide inventory

The importance of having an exhaustive inventory of the mass movement phenomena occurred in the study areas in the past has already been outlined. For this work, the Italian Inventory of Landslide Phenomena (Inventario dei Fenomeni Franosi in Italia, or IFFI) of *Istituto Superiore per la Protezione e la Ricerca Ambientale IdroGEO* was used (32).

The layers taken into consideration from the inventory were the one containing polygonal landslides and the one containing linear landslides.

One of the issues to overcome in this phase was that, while for the polygonal landslides the layer already included the area onto which the mass movement occurs, this is not the case for linear landslides. In the meantime, it is essential to this analysis that the landslide features have a non-zero extension, so that they can be overlapped with the selected terrain factors. The solution to this problem was to approximate the width of the linear landslides to a fixed value: this was the chosen approach because all of the linear landslide features actually belong to the "Debris flow" category, and this type of movement usually takes place in narrow valleys and channels. The value of the width was chosen to be 10 m, based on previous similar studies (Yordanov and Brovelli, 2020a).

These layers, as will be discussed in a following section, were considered when defining the zone containing past landslide phenomena.

5.1.1.3 Factor sampling

The strategy adopted to sample the terrain variables obtained after the preprocessing phase was to create two sets of random points, one distributed in a zone where the probability of having a landslide could be approximated to 0, and one distributed

in the zone where past landslides occurred, so that the landslide probability could be taken as 1. The next paragraphs illustrate the design choices that led to the creation of the two zones and the point sampling process.

Landslide zone

The *Landslide zone* (called *LS zone* in the rest of this thesis) represents the portion of the territory onto which past landslides have occurred. This study focuses on two types of movements: debris flow and rotational/translational slides; therefore, rockfalls and complex landslides were excluded from the polygonal landslides inventory. This filter operation was not necessary for the linear landslides, which all fall under the debris flow category. The remaining features, i.e. polygonal debris flows, polygonal translational/rotational slides and all the linear landslides buffered, were merged, therefore creating the *LS zone*. Table 5.2 contains some statistics about the number of both polygonal and linear landslides included in the *LS zone* for both areas.

Zone	Landslide type	Number of features
Upper Valtellina	Polygonal landslides	637
	Linear landslides	1363
Val Tartano	Polygonal landslides	213
	Linear landslides	441

Table 5.2: Statistics on number and type of landslide features in the *LS zone*

For analytical purposes, this dataset was separated into two sections: the first one, containing 80% of the features, is the LS training dataset, while the second one (20% of the features) is the LS testing dataset. This separation was made in QGIS using its random selection functionality (Figure 5.3 and Figure 5.4).

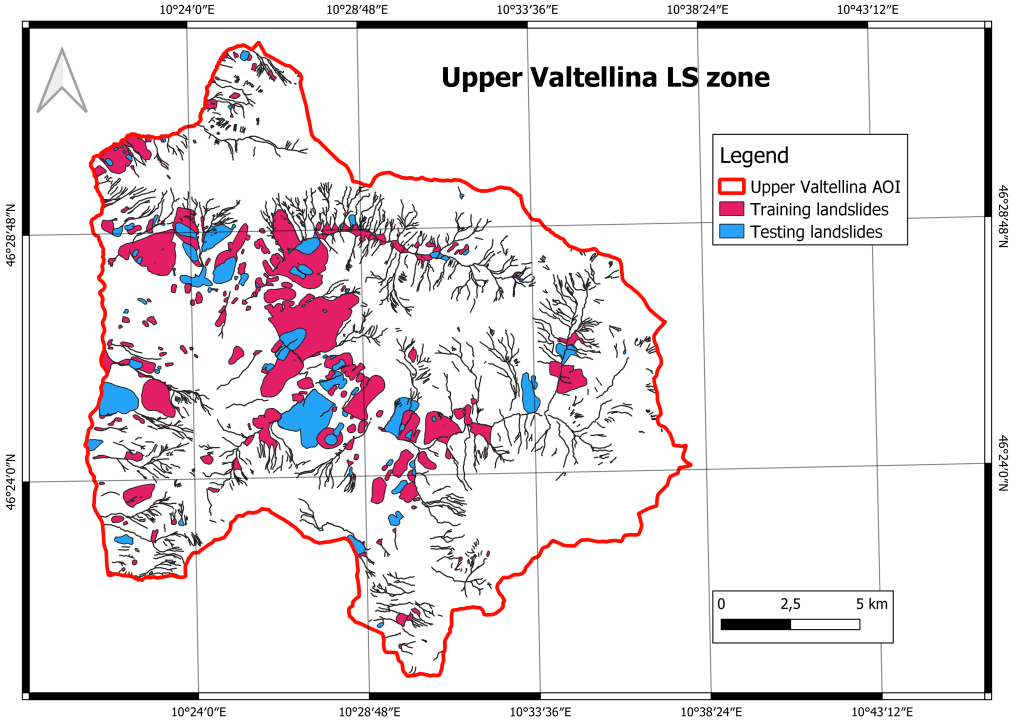


Figure 5.3: Upper Valtellina LS zone

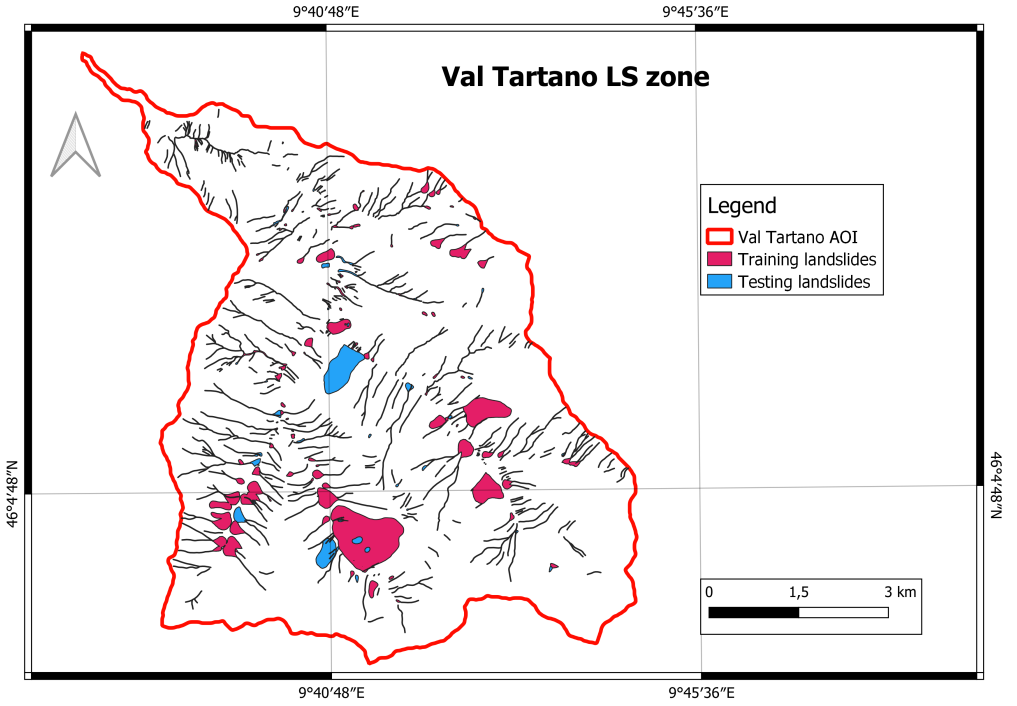


Figure 5.4: Val Tartano LS zone

Val Tartano modification As observed in a previous study (Yordanov and Brovelli, 2020a) it is better to further filter the landslide features of the Val Tartano basin. In particular, upon investigation of the landslides, it was discovered that one landslide (in blue in Figure 5.5), classified as debris flow, actually corresponds more to a debris accumulation. This hypothesis is backed by the fact that the source for the material could be the Pruna landslide (in yellow in Figure 5.5), the biggest landslide in the whole area. Because of this, the feature now identified as a debris accumulation was excluded from the dataset. Moreover, also the aforementioned Pruna landslide was examined, because it is different from all other landslides in the area in terms of extension and of range of elevation (from 550 to 1200 m a.s.l.). Therefore, it could be considered an outlier of our landslide dataset, and it has been seen that it greatly affected the results; the solution found by Yordanov and Brovelli, and adopted in this work, was to exclude the Pruna landslide from the *LS zone*.

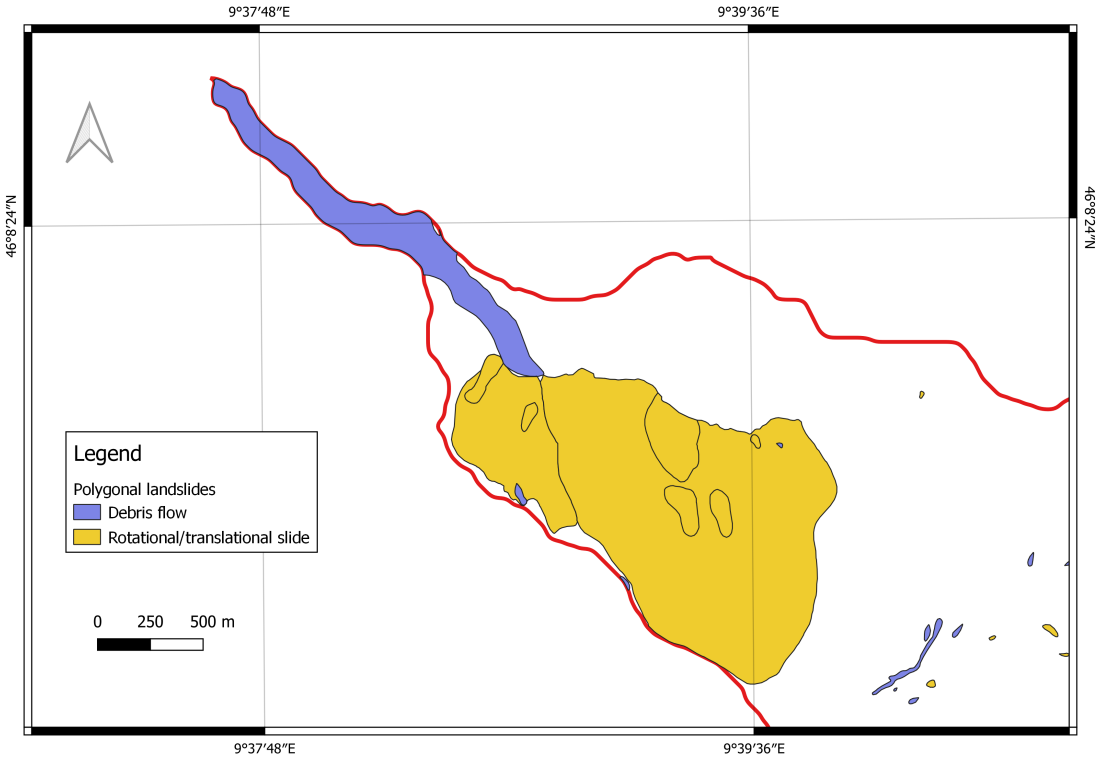


Figure 5.5: Pruna landslide (yellow) and consequent debris accumulation (blue)

No Landslide zone

In this work, along information about the presence of past landslides, the model was also given some information about where landslides are unlikely to occur. To this purpose, a *No Landslide zone* (or *NoLS zone*) had to be defined; also in this case, a previous study (Yordanov and Brovelli, 2020a) was taken into consideration. The study showed that in high landslide density areas such as Val Tartano and Upper Valtellina, the sole hypothesis that the *NoLS zone* was outside of the landslide polygons was not sufficient. This is because the high landslide density makes it so that the landslide training samples end up covering the same range of values of the terrain variables and with the same frequency distribution, leaving us little way to distinguish areas that are prone to mass movements from areas that are not. For this reason, it was decided to use a geological criterion in order to define the *NoLS zone*; in particular, it was hypothesized that in order to be considered *NoLS zone* an area should have a slope angle smaller than 20° or larger than 70°, therefore accounting very smooth or very steep slope. In addition, the terrain lithology was also classified based on the Intact Uniaxial Compressive Strength (IUCS) of the material, a value obtained by tests that describes the maximum stress that a material can take before failing (Table 5.3). With this classification in mind, the *NoLS zone* was created with pixels that both complied with the condition on the slope (slope angle $<$20° or slope angle $>$70°) and where the lithology had an IUCS of 50 MPa or more. To verify the validity of this hypothesis, the pixels of *NoLS zone* overlapping with existing landslides polygons were counted: for both areas, less than 2% of the *NoLS zone* fell onto existing landslides. This overlapping area was later removed, but the hypothesis was considered valid.

Field estimate of Strength	**Examples**	**Strength [MPa]**
Specimen can only be chipped with a geological hammer	Fresh basalt, chert, diabase, gneiss, granite, quarzite	> 250
Specimen requires many blows of a geological hammer to fracture it	Amphibiolite, sandstone, basalt, gabbro, gneiss, granodiorite, limestone, marble, rhyolite, tuff	100 - 250
Specimen requires more than one blow of a geological hammer to fracture it	Limestone, marble, phyllite, sandstone, schist, shale	50 - 100
Cannot be scraped or peeled with a pocket knife, specimen can be fractured with a single blow from a geological hammer	Claystone, coal, concrete, schist, shale, siltstone	25 - 50
Can be peeled with a pocket knife with difficulty, shallow indentation made by firm blow with point of a geological hammer	Chalk, rocksalt, potash	5 - 25
Crumbles under firm blows with point of a geological hammer, can be peeled with a pocket knife	Highly weathered or altered rock	1 - 5
Indented by thumbnail	Stiff fault gouge	0.25 - 1

Table 5.3: Intact Uniaxial Compressive Strength classification

A first computation of the Landslide Susceptibility map for Upper Valtellina with the described definition of the *NoLS zone* was carried out. The resulting map wrongly classified the city of Bormio and the village of Santa Caterina di Valfurva as high risk zones (Figure 5.6). This was because, even though these areas have a very low slope, the underlying lithology material has a IUCS beneath 50 MPa; therefore, they were not included in the *NoLS zone*. In order to correct this anomaly, the hypothesis on the *NoLS zone* was revised to always include areas with very low (i.e. less than 5°) slopes (e.g. cities). The new condition was formulated like:

$$(slope < 5°) \vee [(5° < slope < 20° \vee slope > 70°) \wedge (IUCS > 100\,MPa)] \quad (5.1)$$

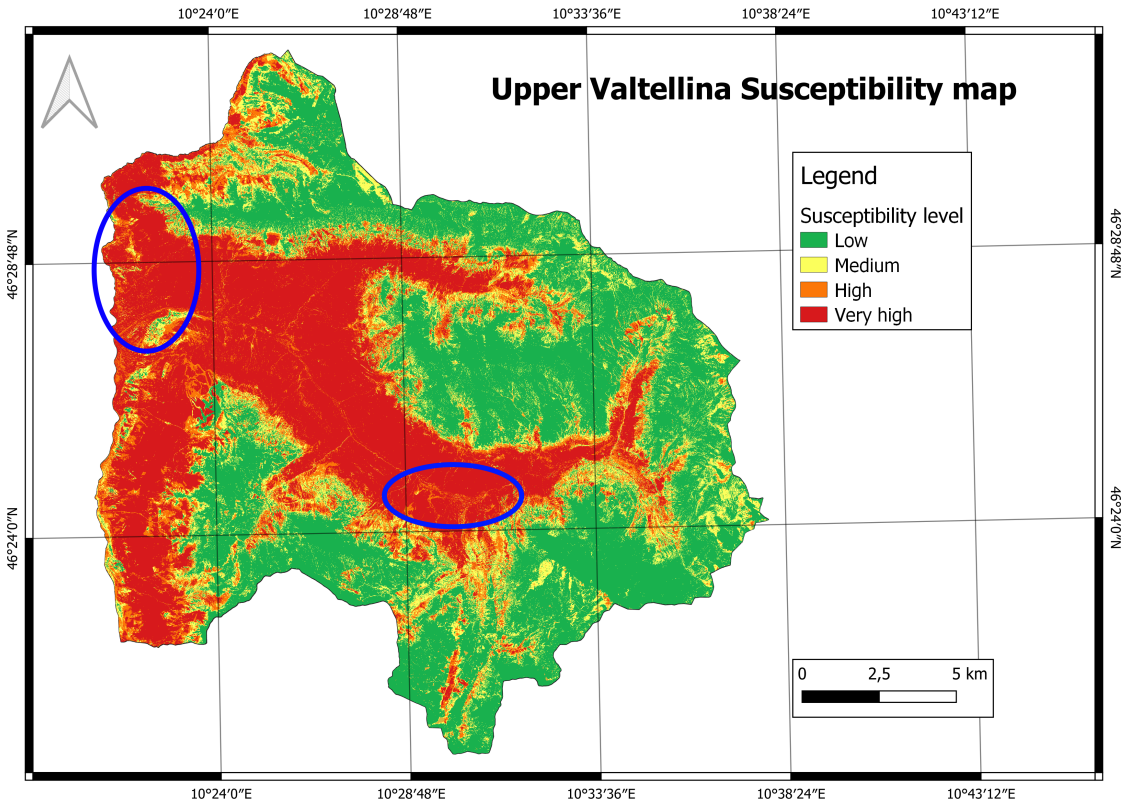


Figure 5.6: LSM for Upper Valtellina with first hypothesis on NoLS zone

The threshold for the IUCS was increased in order to make this new condition valid in terms of overlap with the existing landslides. With this new formula, 1.7% of

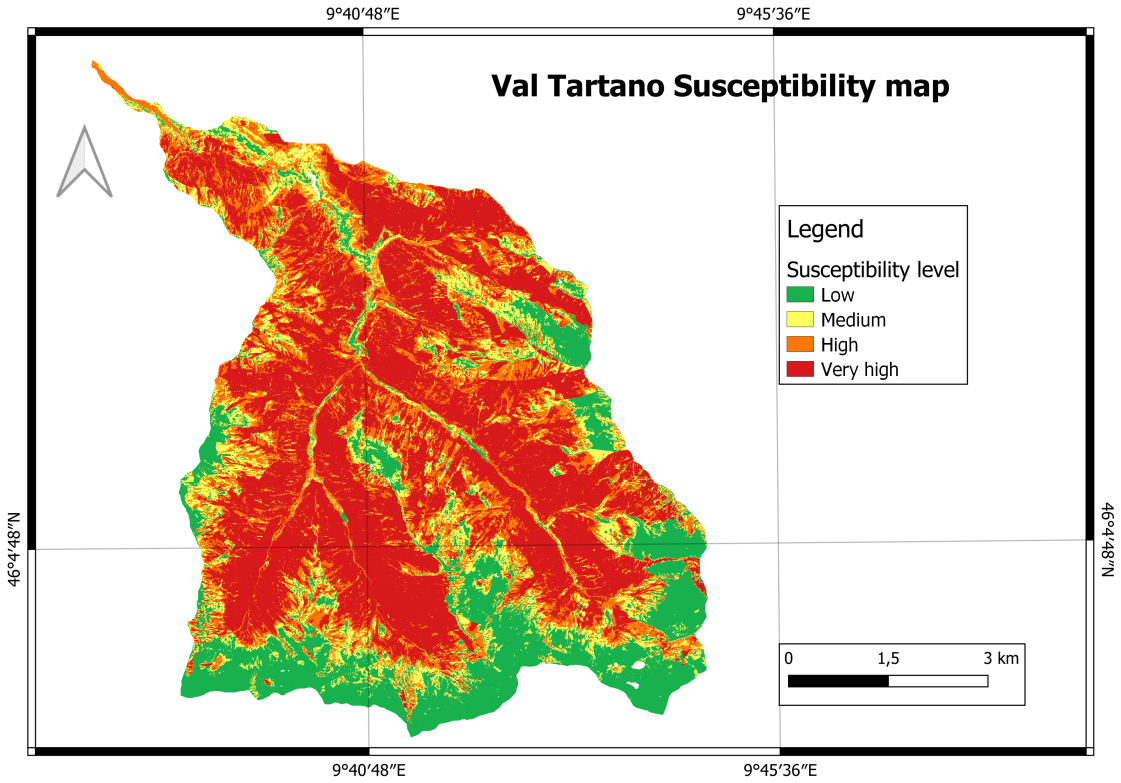


Figure 5.7: LSM for Val Tartano with first hypothesis on NoLS zone

the *NoLS* pixels overlap with past landslides for Upper Valtellina, and 0.5% for Val Tartano. The overlapping pixels were examined, with the purpose of searching for common patterns. No clear pattern was discovered for Val Tartano. On the other hand, it was discovered that for Upper Valtellina the vast majority of excluded pixels (19000 pixels on a total of 32000) belonged to areas with slope smaller than 5°, but with a lithology of deposits or detritical layer. This datum shows that some of the debris mass movements with very gentle slope were included in the *NoLS zone* by hypothesis.

In any case, these overlap areas are very few, thus confirming the *NoLS* hypothesis, and they were removed from the *NoLS zone* before carrying out the analysis. The final maps of the *NoLS zones* can be seen in Figure 5.8 and Figure 5.9.

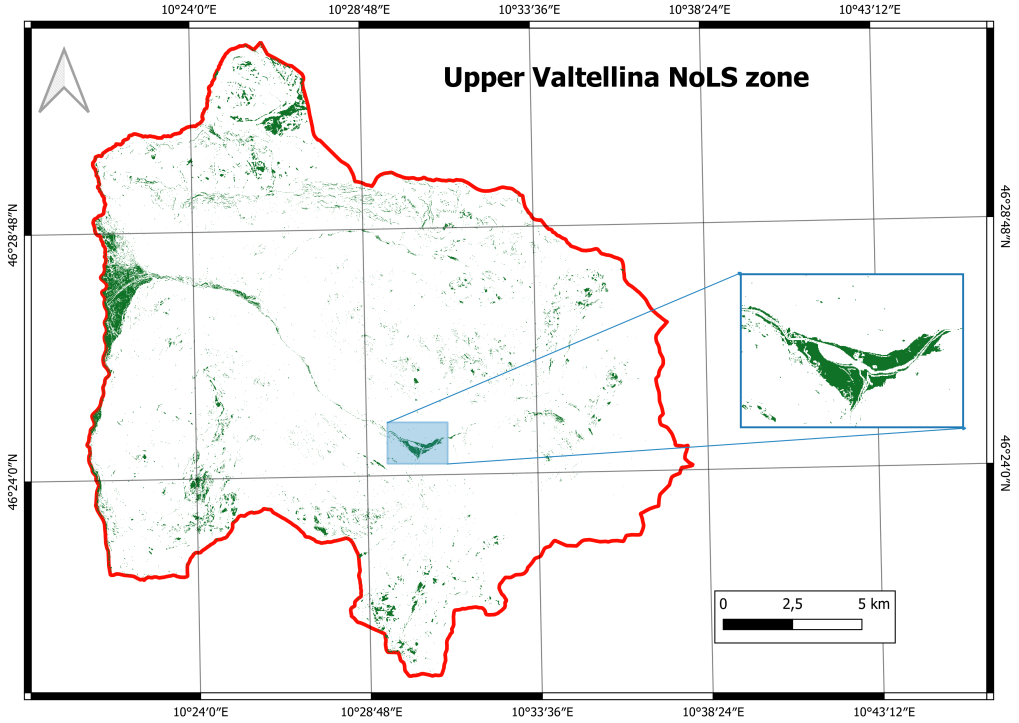


Figure 5.8: Upper Valtellina NoLS zone

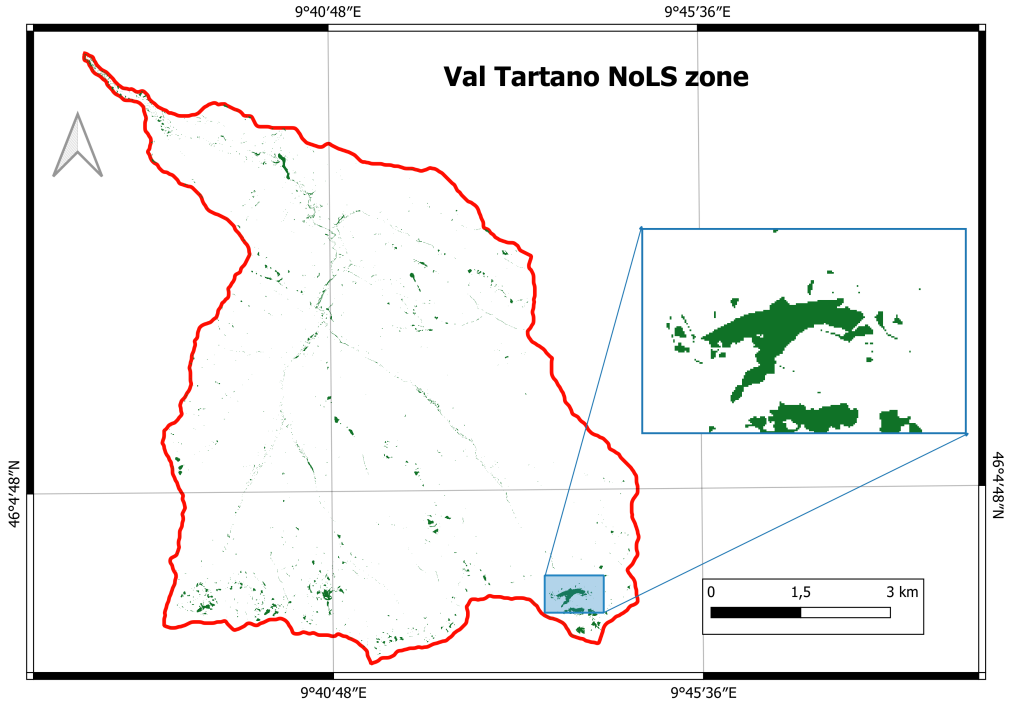


Figure 5.9: Val Tartano NoLS zone

It is also important to note that, since the slope and the lithology layers were exploited for the definition of the *NoLS zone*, they were excluded from the factors considered in the analysis. This decision is motivated by the fact that the considered points, and therefore the results, are already dependent on the two layers. Including them would produce maps with a high prevalence of these factors with respect to the other ones, therefore they were omitted.

Point sampling

The last preparation step is to sample the sets of random points with the chosen terrain variables. It was chosen to sample 100000 points for the Upper Valtellina AOI, and 20000 points for Val Tartano, since it is a smaller area (about 6 times smaller than Upper Valtellina). The function *Random points inside Polygons*[1] of QGIS was used to create the set of points; this function takes as input a polygon layer, a number of points and optionally a minimum distance between points, and outputs a layer containing random points inside the input polygon. The points are chosen with a random distribution using the *random.random()* Python function. If a minimum distance was specified, the points are chosen in such a way that the distance between two points is never less than this parameter. For this analysis, the minimum distance was set to be 5 m, i.e. the chosen mapping unit, in order to avoid points to be too close to one another. The first set of random points was taken in the training portion of the *LS zone*, and another set was respectively taken in the testing portion of the *LS zone*. As for the *NoLS zone*, the random points were first generated and then randomly divided into the training and testing partitions (80% and 20% respectively).

Then, using the *Point Sampling Tool*[2] plugin of QGIS, these point layers were sampled with the values of the raster factors, thus obtaining the point dataset to use for the analysis. It is important to specify that the obtained layers also contained the information about the level of susceptibility in a point, put to 1 in the case of the *LS zone* points, and to 0 for the *NoLS zone* points.

[1] https://docs.qgis.org/3.16/en/docs/user_manual/processing_algs/qgis/vectorcreation.html#random-points-inside-polygons

[2] https://plugins.qgis.org/plugins/pointsamplingtool/

5.1.2 Processing

As already stated, the Random Forests algorithm was chosen for the analysis, and it was implemented in RStudio using the *ModelMap* package. Using this package it is possible to include in a single R function the modelling, validation and mapping functionalities. Besides constructing a predictive model, *ModelMap* can also validate the model with a test set, using Out Of Bag (OOB) predictions. In Random Forests, OOB means that each tree uses the data left out by its random selection to validate its predictions (Breiman, 2001). The code also creates graphs and tables of the model validation diagnostics, and finally *ModelMap* can apply the model to GIS images in order to create prediction maps (Freeman et al., 2016). For investigating the code, please refer to B.1.3.

5.2 Detection of landslide displacements

The purpose of this part of the study is to create a landslide movement detection procedure capable of computing both the direction and the magnitude of pixel movements using Sentinel-2 images and free and open-source software.

The general strategy employed in this work for obtaining landslide displacements in terms of direction and magnitude is to apply a local cross-correlation analysis on a multitemporal images stack. This was achieved using GRASS GIS (OSGeo, 2021) and custom Python scripts. The reasoning behind the preference of GRASS GIS among other GIS applications was mainly the high synergy between GRASS and Python programming, that allows to combine in a single Python script the functionalities and image processing capabilities offered by GRASS functions with the power and flexibility of the Python programming language. In particular, a set of scripts was developed in order to carry out each of the processing steps, which are illustrated in the next sections, in a standalone way.

5.2.1 Data preprocessing

Image preprocessing (B.2.1) Preprocessing steps are applied to the Sentinel-2 Level-1C imagery, collected during the download phase, with the aim to produce a suitable multi-temporal stack. The preprocessing procedure is composed by:

- AOI setting: the AOI is imported (*v.in.ogr*[3]) and the GRASS computational region is set to the extent of the AOI (*g.region*[4]). It is important to note that the AOI selected at this step is larger than the final one, for obtaining better results in the next phase;
- Image import: the folders containing Sentinel-2 images are imported one by one in a Python *for loop* using the *i.sentinel.import*[5] function. Not all bands are imported: in particular, the 60 m original resolution bands, i.e. aerosol (band 1), water vapour (band 9), and cirrus (band 10) are excluded;
- Cloud masking: Sentinel-2 images folders also contain a vector shapefile describing the mask of the clouds covering the image. This vector is imported and rasterized (*v.to.rast*[6]) and then it is used to set any pixel covered by clouds to no-data (*r.mapcalc*[7]);
- Atmospheric correction: the Dark Object Subtraction (DOS) correction is applied to each band using *r.univar*[8] and *r.mapcalc*. This procedure consists in searching for the darkest pixel value in each band, and then subtracting this value from every pixel in the band;
- Output image generation: the bands are grouped in a single virtual raster layer (*i.group*[9]). This layer is then exported as a multiband `.tiff` image (*r.out.gdal*[10]).

Image co-registration (B.2.2) When analysing images in order to detect changes between them, a fundamental step is spatially co-registering the images. Co-registration means to ensure that the images become spatially aligned so that any feature in one image overlaps as well as possible its footprint in all other images in the stack. To do so, the AROSICS (Automated and Robust Open-Source Image Co-Registration Software) package was used. In particular, the global co-registration functionality

[3] https://grass.osgeo.org/grass78/manuals/v.in.ogr.html
[4] https://grass.osgeo.org/grass78/manuals/g.region.html
[5] https://grass.osgeo.org/grass78/manuals/addons/i.sentinel.import.html
[6] https://grass.osgeo.org/grass78/manuals/v.to.rast.html
[7] https://grass.osgeo.org/grass78/manuals/r.mapcalc.html
[8] https://grass.osgeo.org/grass78/manuals/r.univar.html
[9] https://grass.osgeo.org/grass78/manuals/i.group.html
[10] https://grass.osgeo.org/grass78/manuals/r.out.gdal.html

was applied taking the first chronological image as *master* and co-registering all other images with respect to it. The global co-registration computes sub-pixel shifts in the X and Y direction between the two images and corrects the *slave* image, thus aligning it to the *master* (Figure 5.10b).

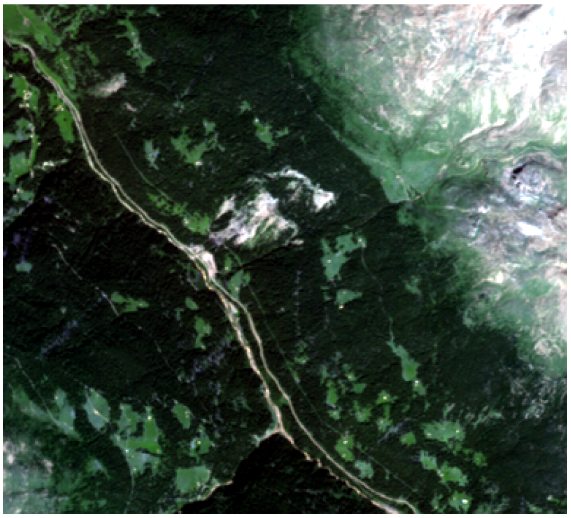
In initial trials, the co-registration process was applied to images with different Relative Orbit numbers: the results were not particularly accurate, and since shifts between the images could prejudice the quality of the displacement analysis, it was decided to consider images with the same Relative Orbit number, in particular Orbit 22.

Clip to the final AOI (B.2.3) A larger AOI was useful for obtaining better results in the co-registration phase. For a better focus on the Ruinon landslide and for speeding up the computation the co-registration image outputs are clipped to a final, smaller AOI (Figure 5.10c).

Histogram matching (B.2.4 and B.2.5) The images' histograms are then matched: this means to transform one image so that the cumulative distribution function (CDF) of values in each band matches the CDF of the corresponding band in another image. In other words, we want to make the *slave* image band values as similar as possible to the ones of the *master* to enhance the cross-correlation between them. The histogram matching was developed entirely in Python exploiting the Scikit image library, and in particular the *match_histograms*[11] function. The single matched bands were later combined together by importing them in GRASS, grouping them and then exporting the multiband image (Figure 5.10d).

[11] https://scikit-image.org/docs/dev/api/skimage.exposure.html#skimage.exposure.match_histograms

(a) Original Sentinel-2 image (zone of the Ruinon landslide in red)



(b) Output of initial preprocessing and co-registration phase

(c) Output after clipping to final AOI

(d) Final preprocessed image after histogram match

Figure 5.10: Preprocessing phases of Sentinel-2 images

5.2.2 Processing

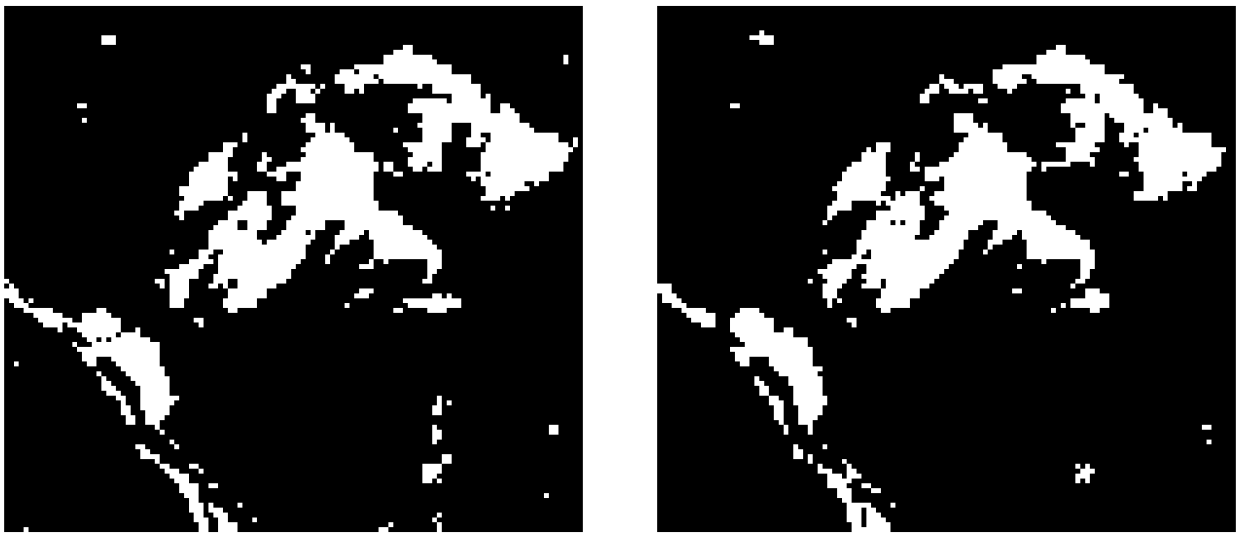
Maximum Cross-Correlation (B.2.6) Once the images have been prepared as described, a local cross-correlation procedure implemented in Python is applied to *master-slave* couples of images. As a first step, the user can define the size of the moving window, which is the subset of the input maps that will be compared iteratively between the *master* and the *slave* (You et al., 2017). Different trials were carried out varying the size of the moving window, and the optimal window size was found to be 7x7 pixels. Then, a loop iterates placing the window on a pixel of the *master* and on the corresponding pixel of the *slave* (i.e. the pixel at the same coordinates). At this point, the maximum cross-correlation process is carried out on the two windows using the *phase_cross_correlation*[12] function of the *scikit registration* package. This function outputs the X and Y shifts (in pixels) required to register the window of the *slave* with the one of the *master*. The code later moves the window on the *slave* according to the computed shifts and computes the center pixel of the new window: this is the pixel correlated to the center of the window in the *master*. At this point, the distance and the angle between these two points are computed, and these values are stored into outputs raster maps at the position of the central pixel on the *master* image. The normalized root mean square correlation error (from 0 to 1), which is a metric of the reliability of the computed cross-correlation, is also stored in an output map. This computation is carried out for each pixel on the images by moving the windows by 1 pixel in each iteration of the *for loop*. Finally, the outputs of the algorithm are three maps for each *master-slave* couple, containing the displacement magnitudes (in pixels), the displacement directions (in degrees from North) and the root mean square error for each pixel whose movement was detected.

This whole procedure was carried out both considering a *fixed master*, i.e. the *master* is fixed to the first chronological image and all the other images are coupled with it, or a *moving master*, i.e. the *master* is always the previous chronological image with respect to the *slave*.

[12] https://scikit-image.org/docs/stable/api/skimage.registration.html#skimage.registration.phase_cross_correlation

5.2.3 Preliminary experiments

For developing the described procedure, inspiration was taken from the work of Oxoli et al. (2020), that used a similar strategy for the detection of desert dunes displacements. The approach used by Oxoli et al. included an unsupervised classification step, in order to divide the territory into two zones: *dunes* and *not dunes*. This approach was considered also in the case of the Ruinon landslide, developing an unsupervised classification procedure in order to distinguish "landslide" pixels from "no landslide" pixels; after a few trials, it was seen that this process introduced too much differences between the images, given the fact that along the landslide body the trees and the bare soil usually merge and the classification struggles to consistently classify them (Figure 5.11). This inconsistency was detrimental for the sake of the cross-correlation, thus the procedure was reworked to be able to handle directly multiband images.



(a) Classified image of 18/07/2019 (b) Classified image of 23/07/2019

Figure 5.11: Comparison between two similar images (only 5 days apart) highlighting differences on the border and inside the body of the landslide

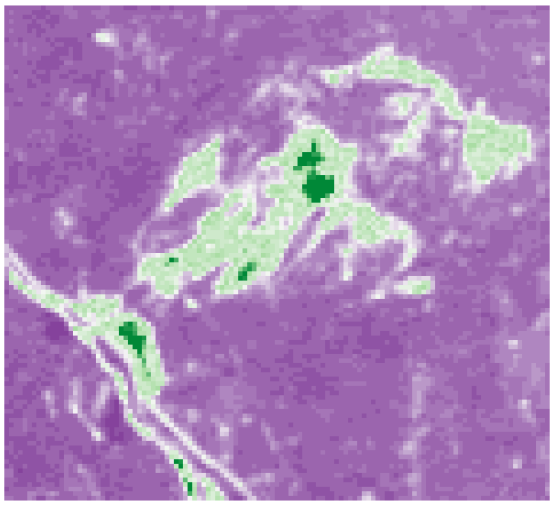
Another approach that was explored was to use images with a particular band combination as input for the procedure. In particular, three indices were calculated for each preprocessed image:

$$Bare\ Soil\ Index = \frac{(SWIR + Red) - (NIR + Blue)}{(SWIR + Red) + (NIR + Blue)}$$

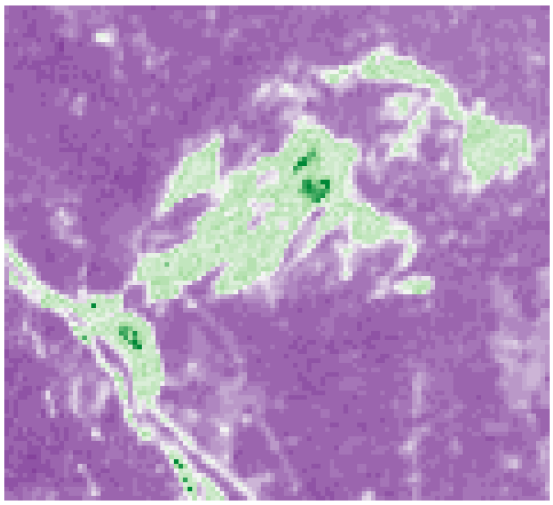
$$Normalized\ Difference\ Vegetation\ Index = \frac{NIR - Red}{NIR + Red}$$

$$Normalized\ Difference\ Water\ Index = \frac{NIR - SWIR}{NIR + SWIR}$$

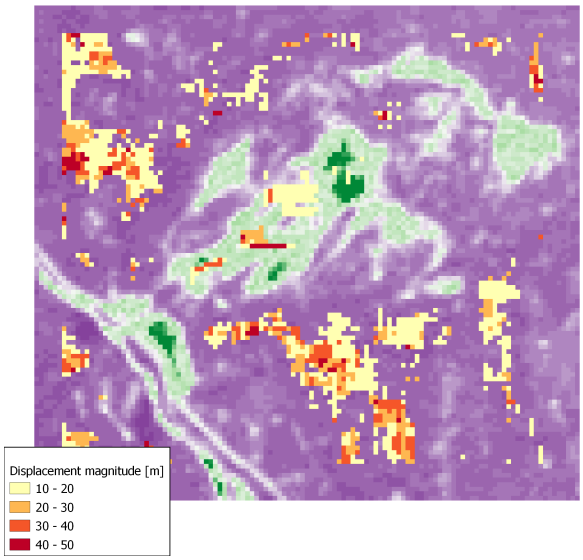
where $SWIR$ stands for Short-Wave InfraRed wavelength and NIR for Near InfraRed wavelength. The BSI, NDVI and NDWI were later assigned respectively to the Red, Green and Blue bands of the input images to be used for cross-correlation. However, even if this band combination clearly highlights the landslide body with respect to the rest of the territory, it also introduces a lot of variability in the values of the pixels, both in the landslide body and outside of it, as seen in Figure 5.12c. Therefore, also this method was disregarded, and in general it was decided to apply the procedure directly on RGB images, without any modification that could introduce errors.



(a) Image of 27/08/2019 with band combination applied



(b) Image of 21/09/2019 with band combination applied



(c) Output of the operation

Figure 5.12: Analysis between August and September 2019 with images modified using the BSI, NDVI and NDWI indices as bands

After the whole method was refined to work directly on RGB images, disregarding classification and band combination, an experiment was carried out to verify the performances of the cross-correlation. Since ground measurements on the Ruinon

landslide were obtained by concession of ARPA Lombardia, some of them were used to validate the results obtained by the cross-correlation. In particular, multitemporal topographic measurements for a set of targets positioned on the body of the landslide (called *mire ottiche*) were considered. All the data available were filtered by considering only measurements from 2015 onwards (since the Sentinel-2 mission was launched that year), thus obtaining a set of 22 *mire ottiche* (Figure 5.13). These data were analysed with a GIS software, specifically looking for a displacement of at least 10 m between two consecutive epochs: this is because the cross-correlation procedure is able to identify measurements with a magnitude of minimum 1 pixel, i.e. 10 m for Sentinel-2 images.

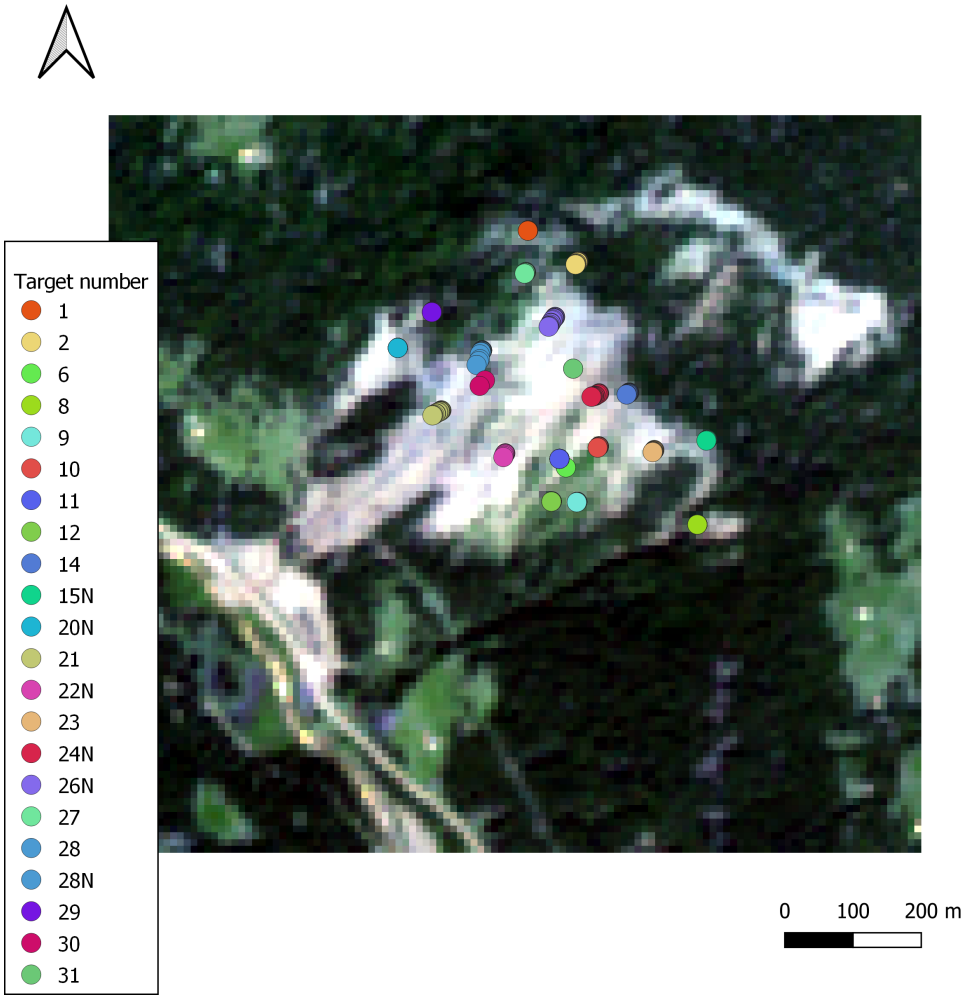


Figure 5.13: All available data from *mire ottiche* from 2015 to 2020. A target contains one point for each day an observation has been registered

Only one target that satisfied said properties was found, in particular *mira ottica* number 30 showed a displacement of 11.23 m in the period between 18/05/2020 and 25/08/2020 (Figure 5.14). To verify if this displacement was identified by the cross-correlation process, the procedure was applied to two Sentinel-2 images covering the same period: the first one was taken on exactly 18/05/2020 and the second one was taken on 26/08/2020. The cross-correlation procedure was carried out between these two images, and the output maps identified a movement in the pixel containing the starting position of the target. This movement had a magnitude of 10 m, and a South direction (Figure 5.15). This was considered a satisfactory result, especially regarding the magnitude of the displacement.

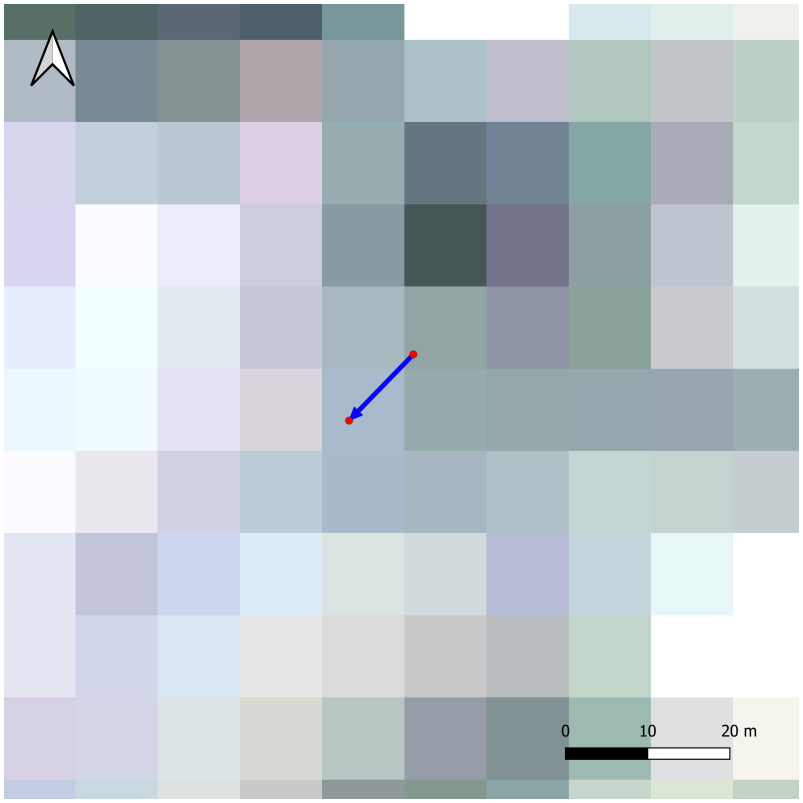
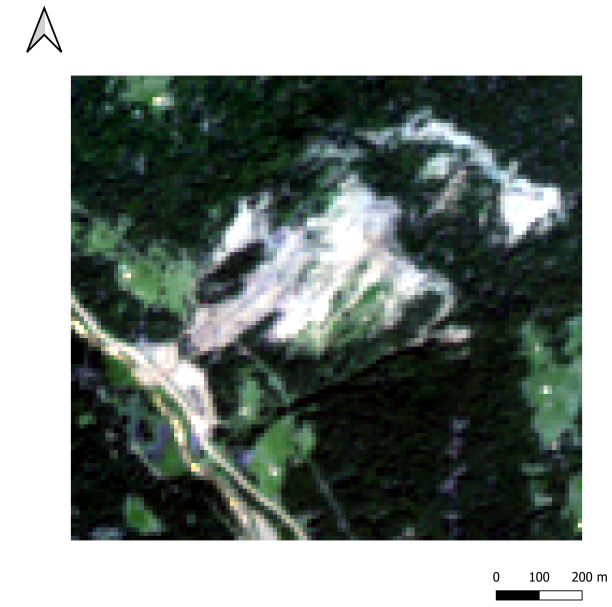
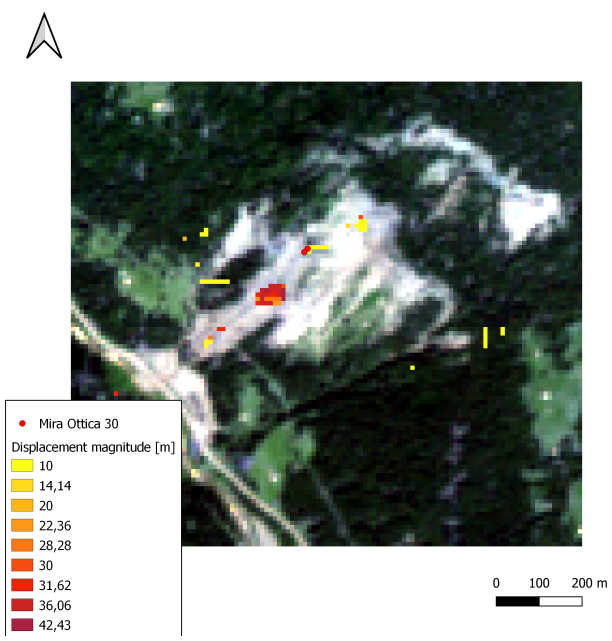


Figure 5.14: Mira ottica number 30

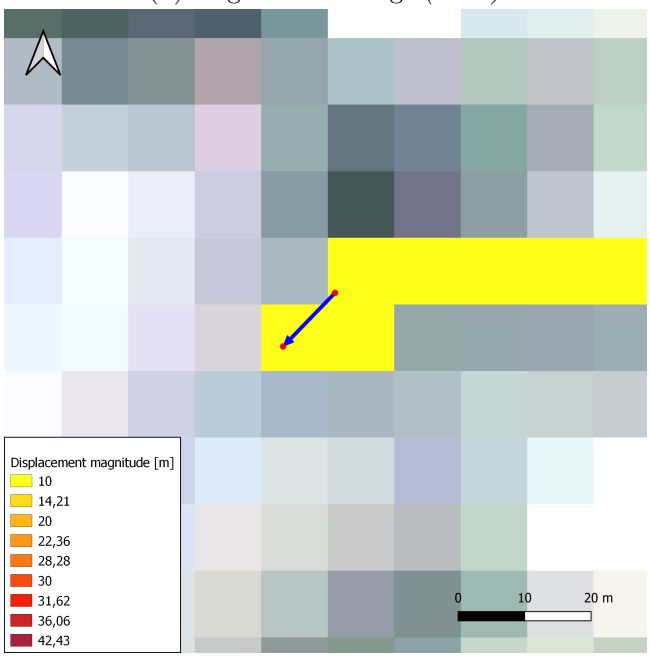


(a) May 2020 image *(master)*

(b) August 2020 image *(slave)*



(c) Displacement magnitude output



(d) Zoom on the mira ottica

Figure 5.15: Results of the preliminary landslide displacement analysis

Chapter 6

Results

6.1 Landslide susceptibility analysis

6.1.1 Upper Valtellina

6.1.1.1 Output

The output susceptibility map for Upper Valtellina (Figure 6.1) correctly identifies the urbanized area and the valley bottom in the lower susceptibility bracket. Moreover, it seems that the areas with a higher risk of landslides are mainly located along the lower sides of the valley, while the high mountain peaks have a low susceptibility level. Table 6.1 shows the extension of each susceptibility level.

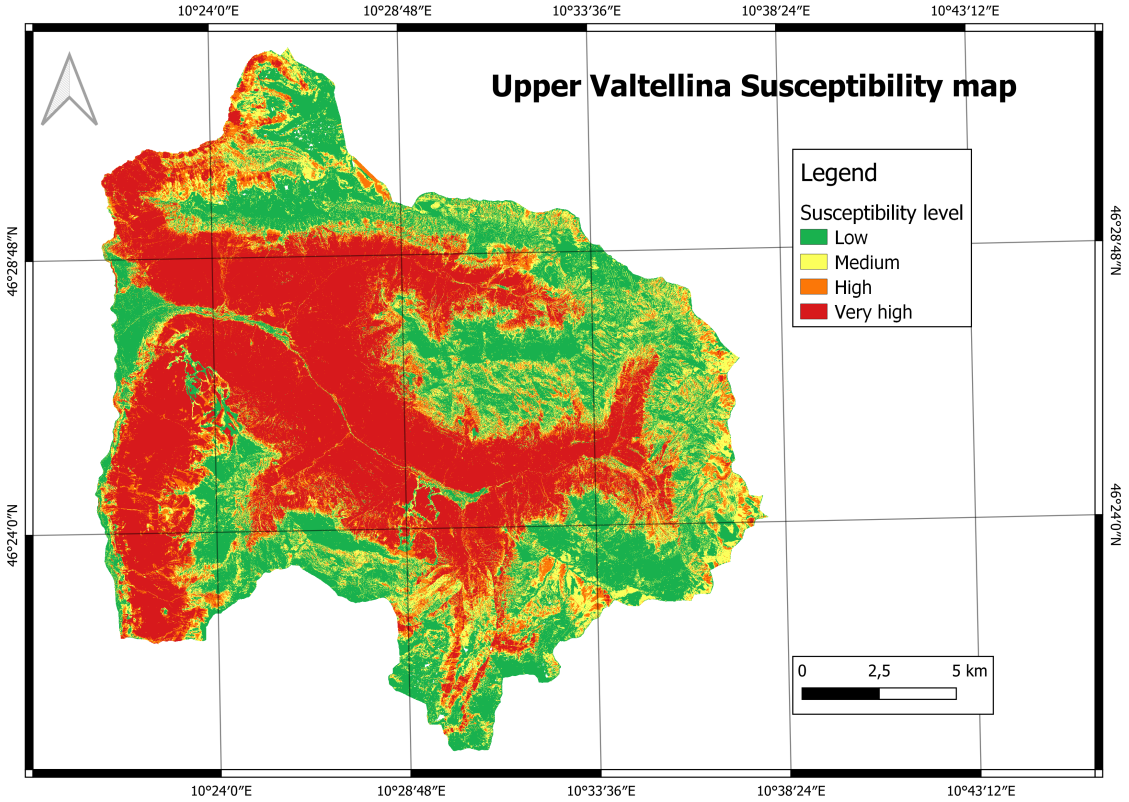


Figure 6.1: Landslide Susceptibility Map for Upper Valtellina

Susceptibility level	Area [km^2]	Area percentage [%]
Low	92.7	33.58
Medium	53.52	19.39
High	37.77	13.68
Very high	92.06	33.35

Table 6.1: Extension of the different susceptibility levels for Upper Valtellina

6.1.1.2 Validation

The model was firstly validated computing its Overall Accuracy (OA), by building a confusion matrix using the testing dataset (Table 6.2). For this purpose, the output map was reclassified and transformed into a binary map, where susceptibility levels from 0 to 0.5 corresponded to "no landslide", and susceptibility levels from 0.5 to 1 corresponded to "landslide".

Reference / Classified	No Landslide	Landslide
No Landslide	9329	1321
Landslide	475	8642

Table 6.2: Confusion matrix for Upper Valtellina obtained with the testing data

The Overall Accuracy score was 90.9%.

To further validate the obtained model, two methods were considered: the Receiver Operating Characteristic (ROC) curve and the Precision Recall Curve (PRC). The area under the curve of the ROC (AUC-ROC) is a widely adopted method to evaluate the performance of classifiers in Landslide Susceptibility modelling. In order to obtain the ROC curve, two parameters are needed: the *sensitivity* of the model (or *true positive rate*) and the *specificity* of the model (or *true negative rate*). They can be calculated using these formulas:

$$sensitivity = \frac{TruePositive}{TruePositive + FalseNegative}$$

$$specificity = \frac{TrueNegative}{TrueNegative + FalsePositive}$$

If we plot these two parameters against each other, we obtain the ROC curve. In order to measure the model performances it is possible to compute the AUC-ROC, i.e. the area of the zone underlying the ROC curve. As a reference, an AUC-ROC equal to 1 means a perfect classification, while an AUC-ROC equal to 0.5 represents a random classification.

The PRC on the other hand is a less common method in LSM, but it has been proven to be as reliable as the AUC-ROC, if not even more reliable in situations of imbalanced datasets (Saito and Rehmsmeier, 2015). It can be computed by plotting the *precision* (Equation 6.1) against the *recall*, which is equal to the *sensitivity*. The advantage of the PRC is that by construction it does not consider the True

Negative parameter, that usually comprises the most numerous part of the outputs in classification models. Because of this the PRC is not affected by any imbalance in the dataset (Yordanov and Brovelli, 2020b).

$$precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{6.1}$$

In the case of Upper Valtellina, the validation results for both the fit and the predictive performances of the model are summarized in Table 6.3 and plotted in Figure 6.2 and Figure 6.3.

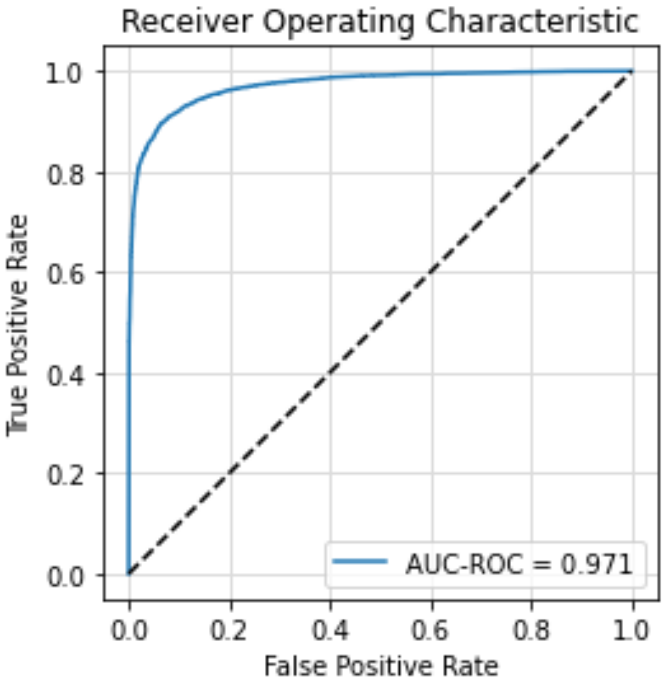


Figure 6.2: ROC curve for Upper Valtellina predictive performances

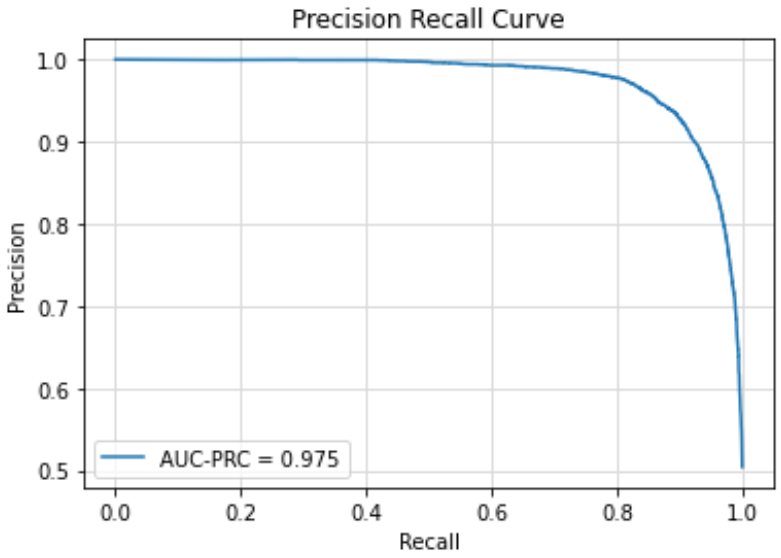


Figure 6.3: Precision Recall Curve for Upper Valtellina predictive performances

Performance type / Validation Method	Model Fit	Model Prediction
AUC-ROC	1.0	0.971
AUC-PRC	1.0	0.975

Table 6.3: Upper Valtellina validation results

As expected, the model fit shows perfect performances, since it is computed with the training data. Furthermore, the computed model shows very good prediction performances, with high results for both the AUC-ROC and the AUC-PRC metrics.

6.1.2 Val Tartano

6.1.2.1 Output

Figure 6.4 shows the output of the model for the Val Tartano AOI, while Table 6.4 shows the extension of each susceptibility level.

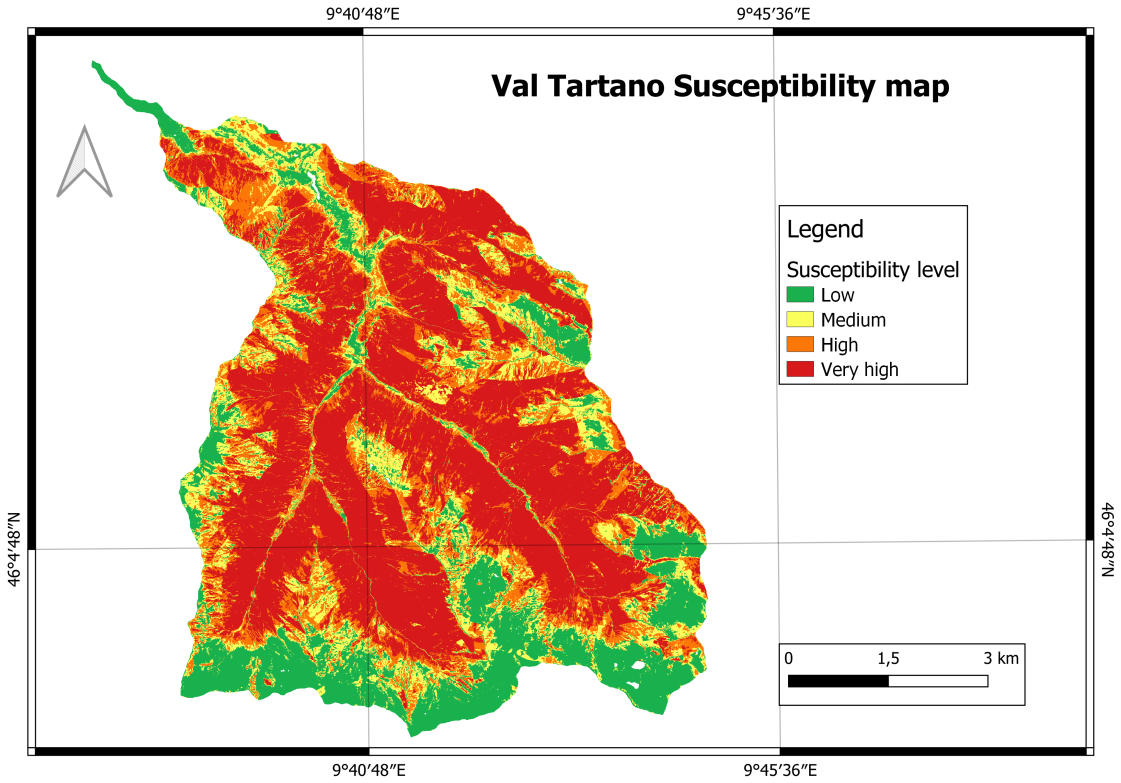


Figure 6.4: Landslide Susceptibility Map for Val Tartano

Susceptibility level	**Area [km²]**	**Area percentage [%]**
Low	9.55	19.17
Medium	7.67	15.41
High	9.96	20
Very high	22.61	45.42

Table 6.4: Extension of the different susceptibility levels for Val Tartano

The first assessment of the quality of this map comes from the fact that the area of the Pruna landslide is correctly classified as high and very high risk, despite being excluded from the *LS zone* (Figure 6.5).

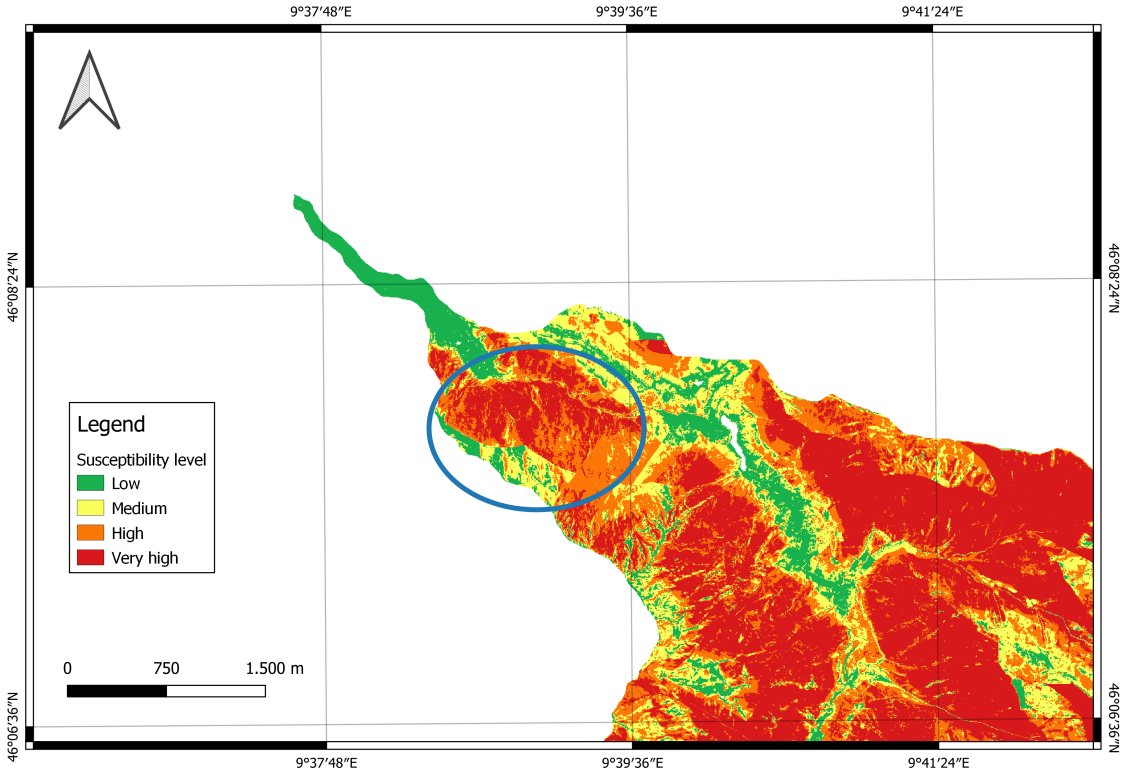


Figure 6.5: Pruna landslide (area in the blue circle) correctly classified

6.1.2.2 Validation

Also in the case of Val Tartano, the model was validated by means of Overall Accuracy, AUC-ROC and AUC-PRC. The Overall Accuracy score in this case was 91.6%. The results of the validation are illustrated in Table 6.5, Table 6.6, Figure 6.6 and Figure 6.7.

Classified \ Reference	No Landslide	Landslide
No Landslide	1800	223
Landslide	104	1777

Table 6.5: Confusion matrix for Val Tartano obtained with the testing data

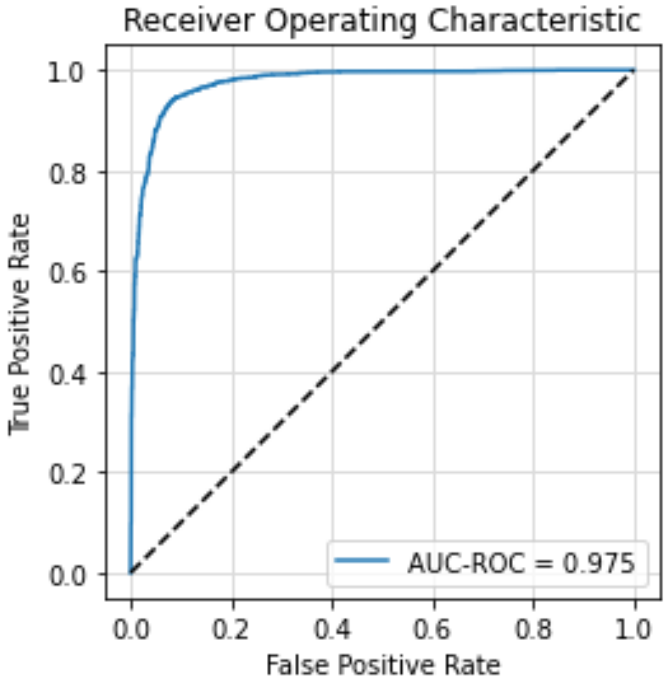


Figure 6.6: ROC curve for Val Tartano predictive performances

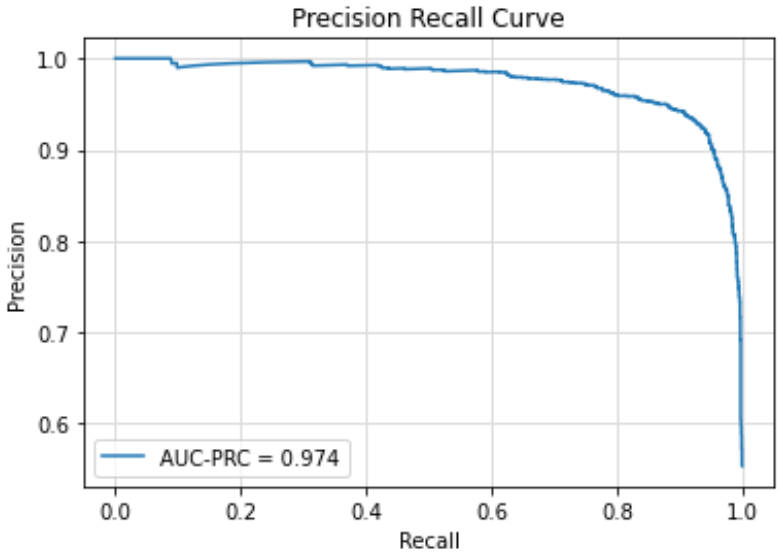


Figure 6.7: Precision Recall Curve for Val Tartano predictive performances

The model shows perfect fit performances and great results in the predictive performances also for Val Tartano.

Performance type / Validation Method	Model Fit	Model Prediction
AUC-ROC	1.0	0.975
AUC-PRC	1.0	0.974

Table 6.6: Val Tartano validation results

Visual validation Moreover, a known landslide (Figure 6.8) shapefile with an extension of 924 m^2, which was not included in the landslide inventory of Val Tartano, was available. This allowed to visually inspect the results of the output map in the area covered by this landslide, in order to check if the results depicted a high risk. As seen in Figure 6.9, the map shows very high susceptibility levels for almost all the pixels overlapping the landslide area, therefore confirming the validity of the results.

Figure 6.8: Known landslide 3D reconstruction from UAV survey

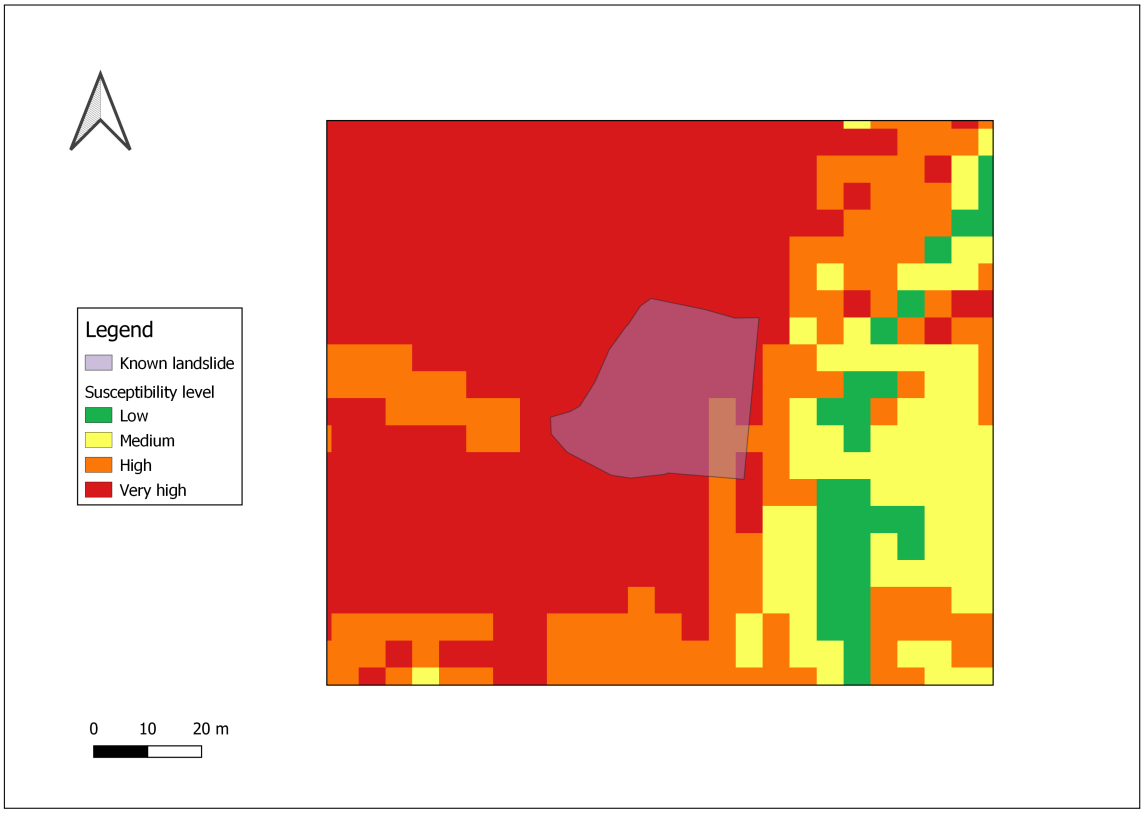


Figure 6.9: Known landslide in Val Tartano correctly classified

6.2 Landslide displacement monitoring

For monitoring the activity of the Ruinon landslide, two different sets of images were considered. The first one consists of one image per year in the period 2015-2020 (Table 6.7), with the idea to track the evolution of the landslide throughout the last few years. The other one is composed by three images, one per month, in the period July 2019-September 2019 (Table 6.8), aiming at highlighting a large movement that took place in the summer of 2019. Both the sets were analysed with moving and fixed master approaches, and the results are illustrated in the next sections.

6.2.1 Yearly frequency

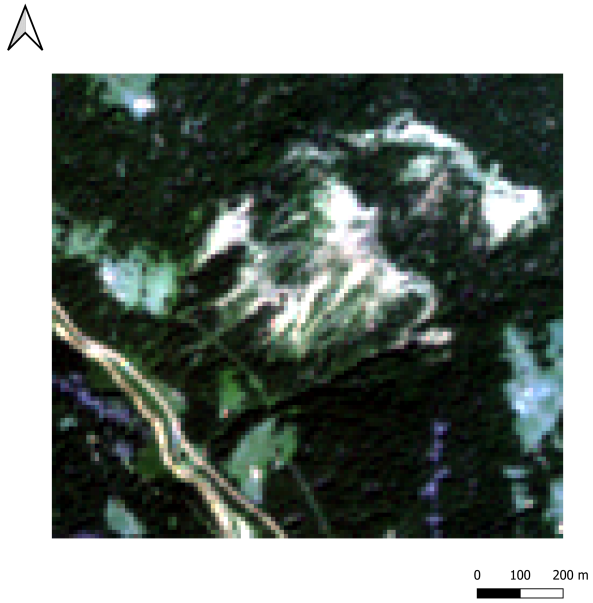
Here are reported the outputs of the displacement monitoring process on images that span from 2015 to 2020. In particular, images in the summer months were chosen, due to the absence of snow. The selected images are highlighted in green

in Table 6.7. After an initial exploration of the available images, the year 2017 was found to have no clear images (i.e. free from clouds and snow); luckily, the landslide did not show large movements from 2016 to 2018, and therefore 2017 was excluded from the analysis.

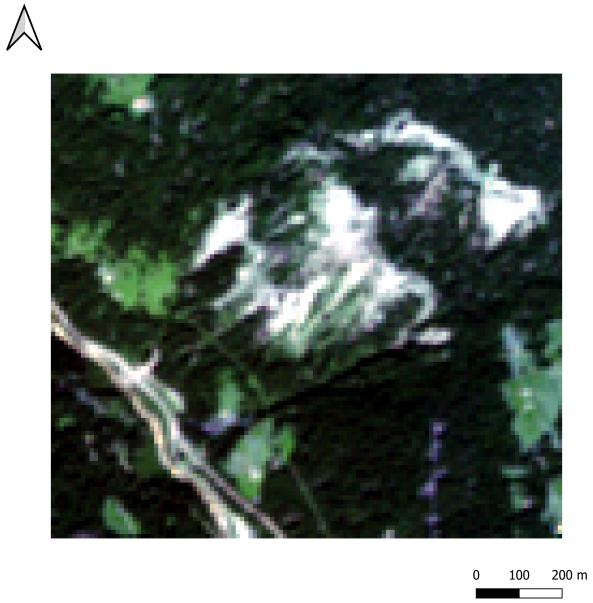
Image code	Sensing date	Selected
S2A_MSIL1C_20150803T101016_N0204_R022_T32TPS	03 August 2015	
S2A_MSIL1C_20160827T101022_N0204_R022_T32TPS	27 August 2016	
S2A_MSIL1C_20170921T101021_N0205_R022_T32TPS	09 July 2017	
S2A_MSIL1C_20180708T101031_N0206_R022_T32TPS	08 July 2018	
S2A_MSIL1C_20190723T101031_N0208_R022_T32TPS	27 July 2019	
S2A_MSIL1C_20200826T101031_N0209_R022_T32TPS	26 August 2020	

Table 6.7: Yearly Sentinel-2 images

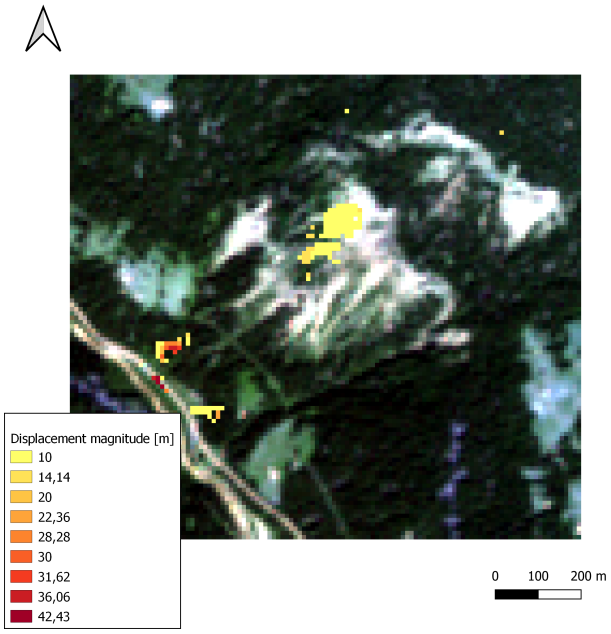
Fixed master The next figures will report the displacement magnitude and direction outputs for every image couple processed with a *fixed master* approach.



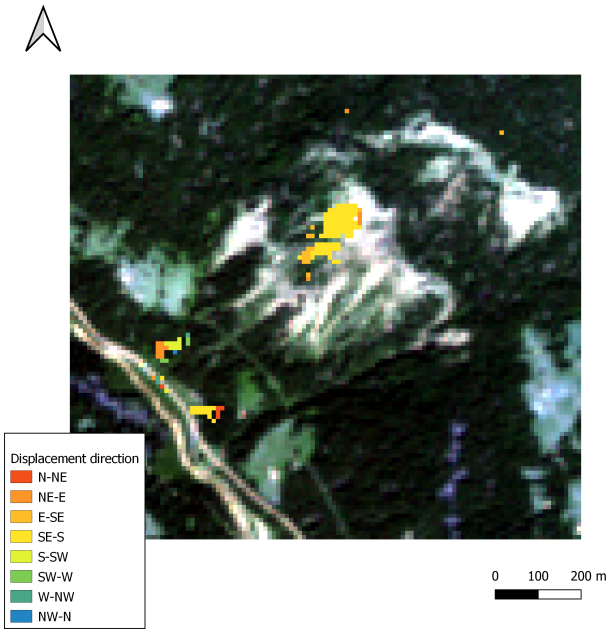
(a) 2015 image *(master)*



(b) 2016 image *(slave)*

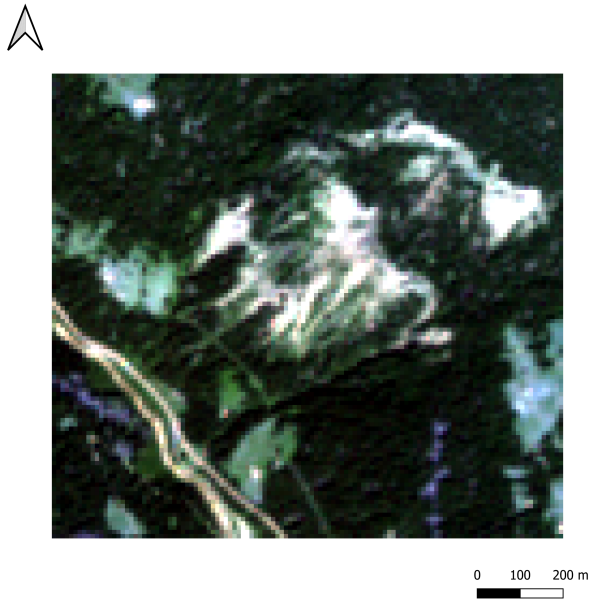


(c) Displacement magnitude output

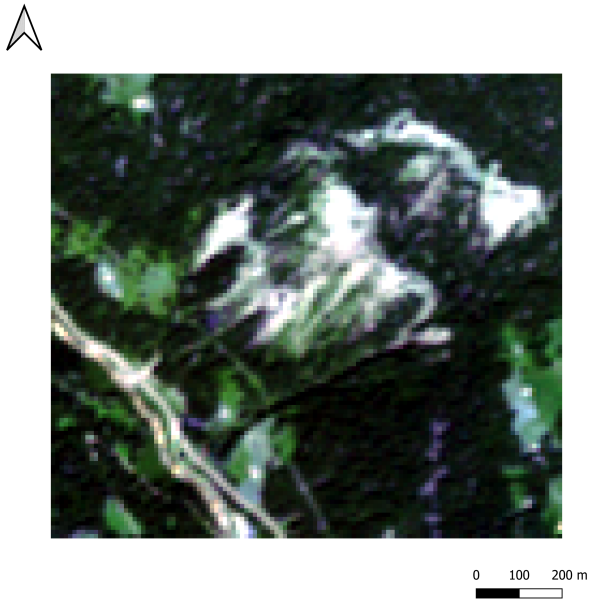


(d) Displacement direction output

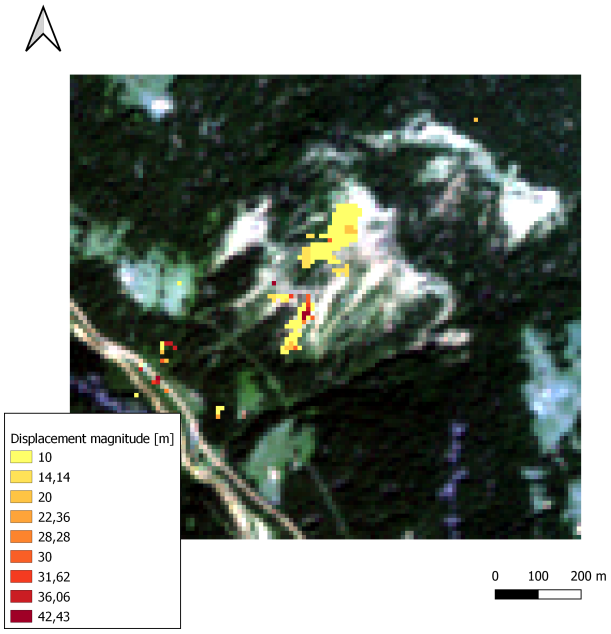
Figure 6.10: 2015-2016 fixed master outputs



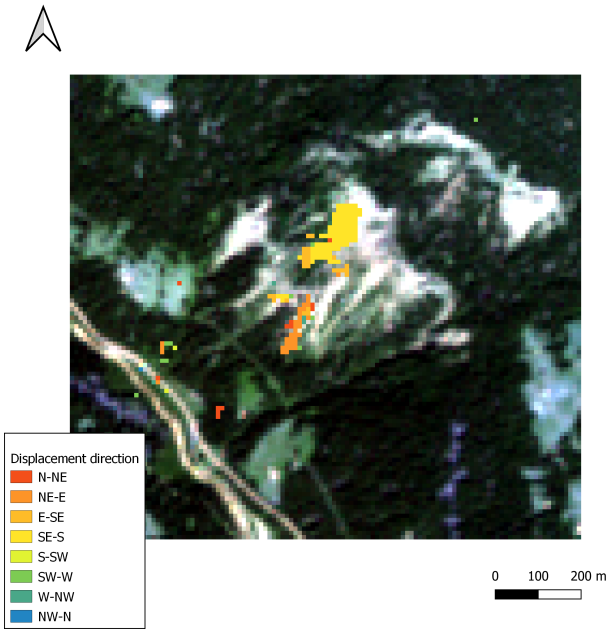
(a) 2015 image *(master)*



(b) 2018 image *(slave)*

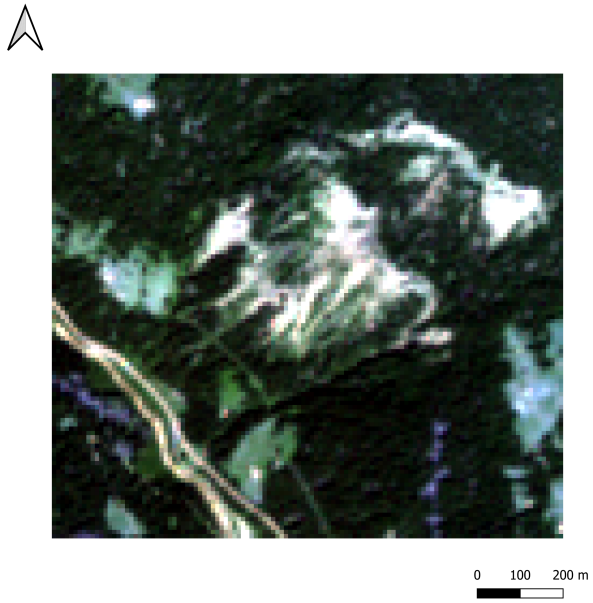


(c) Displacement magnitude output

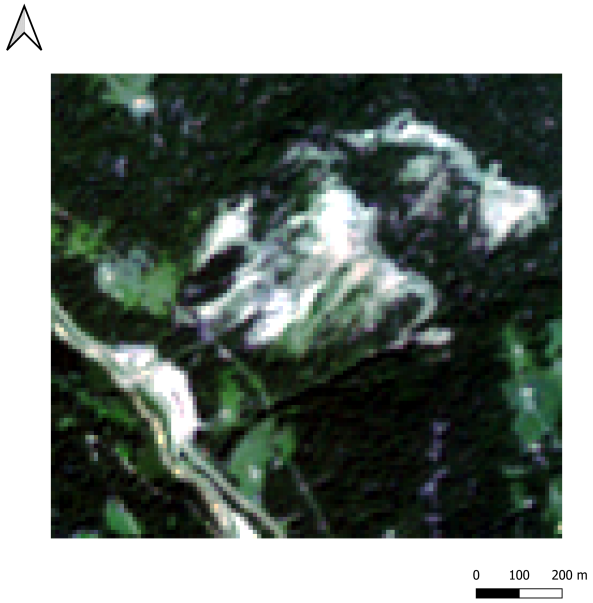


(d) Displacement direction output

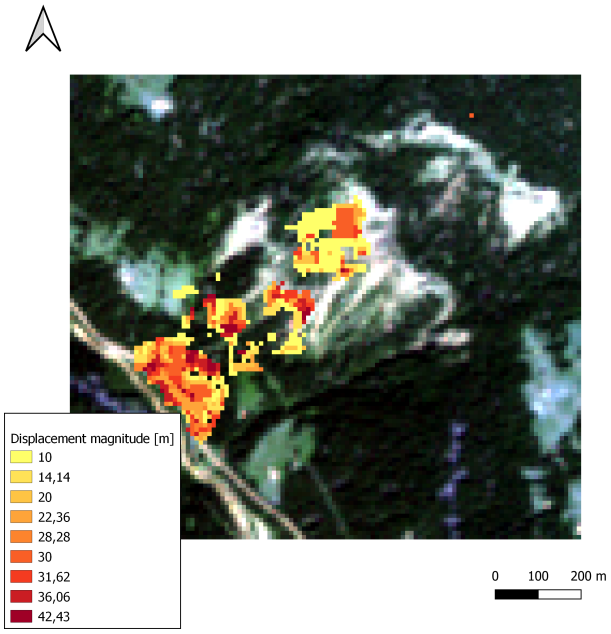
Figure 6.11: 2015-2018 fixed master outputs



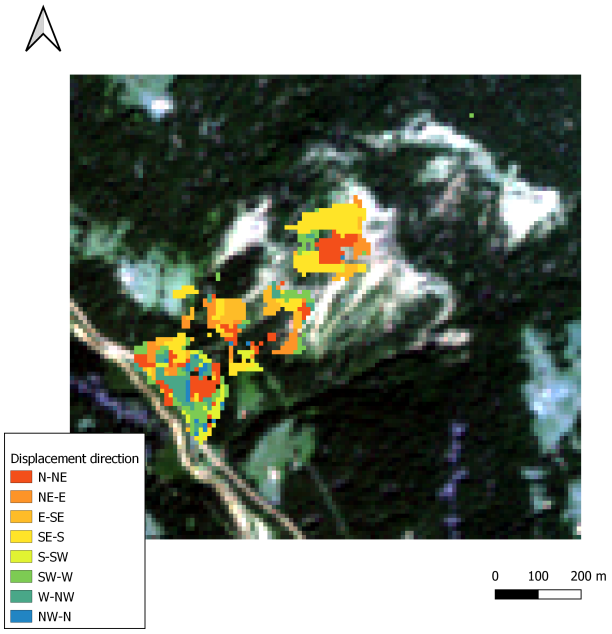
(a) 2015 image *(master)*



(b) 2019 image *(slave)*

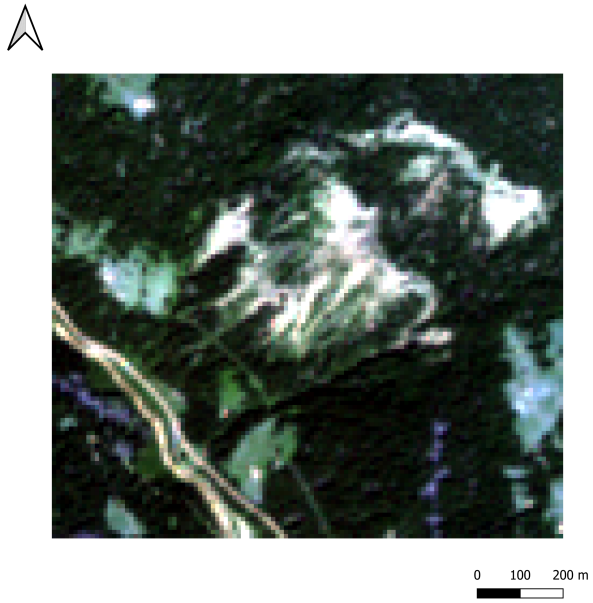


(c) Displacement magnitude output



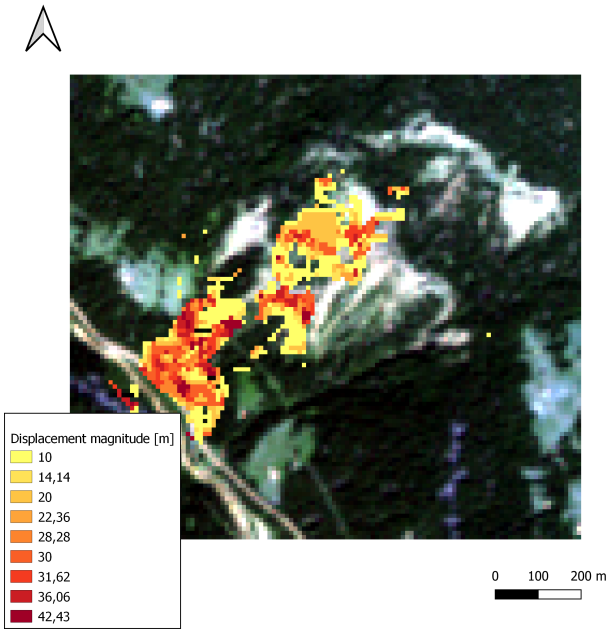
(d) Displacement direction output

Figure 6.12: 2015-2019 fixed master outputs

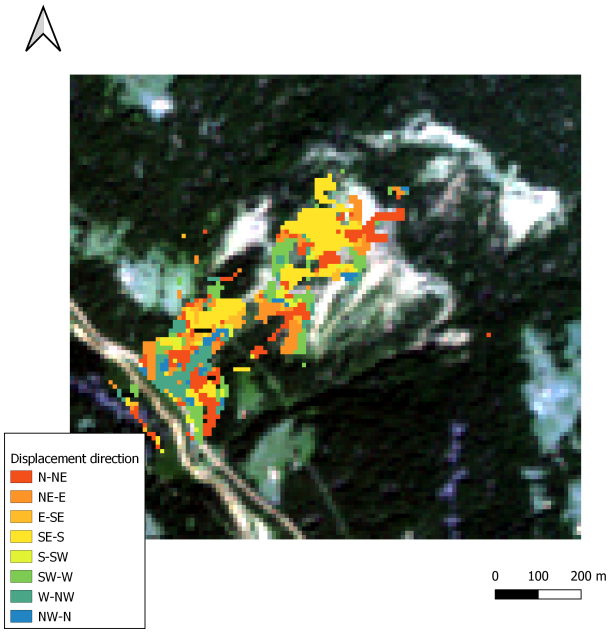


(a) 2015 image *(master)*

(b) 2020 image *(slave)*



(c) Displacement magnitude output



(d) Displacement direction output

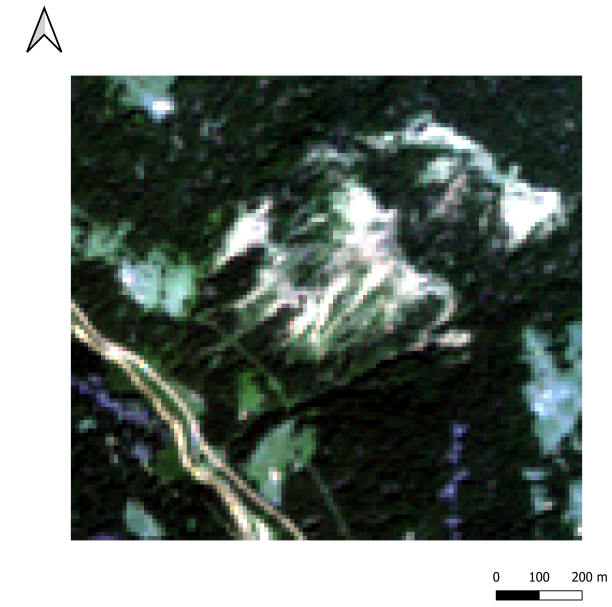
Figure 6.13: 2015-2020 fixed master outputs

By just inspecting the couples of satellite images, it seems that the outputs are correctly identifying zones of the landslide that are moving. We can also notice that, since this is a *fixed master* procedure, the number of pixels identified as moving

is increasing from year to year, and the pixels that are depicted as moving in an image seem to be consistently moving also in the following images. This is a first confirmation of the validity of the results.

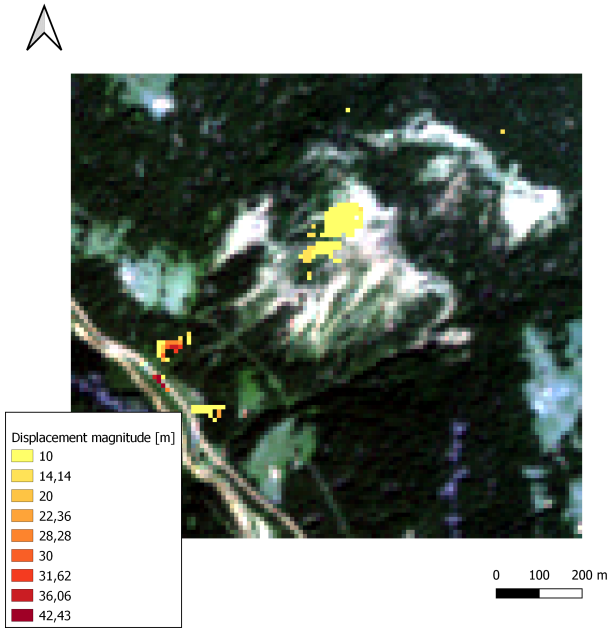
This procedure also highlights changes that are unrelated to the landslide: this is particularly evident in Figure 6.12, where the appearance of a worksite near the road results in the identification of displacements, registered in the output. As it is visible in the figure, the directions of the displacements in the pixels of the worksite have non homogeneous and chaotic values, since the algorithm detects shifts in various directions. This is due to the fact that the algorithm is cross-correlating the forest that was cut down to build the worksite with the forest all around it.

Moving master The next figures will report the displacement magnitude and direction outputs for every image couple processed with a *moving master* approach.

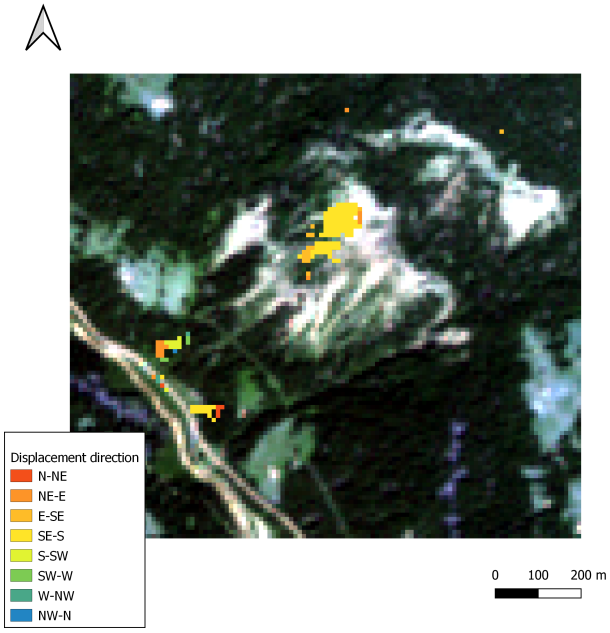


(a) 2015 image *(master)*

(b) 2016 image *(slave)*

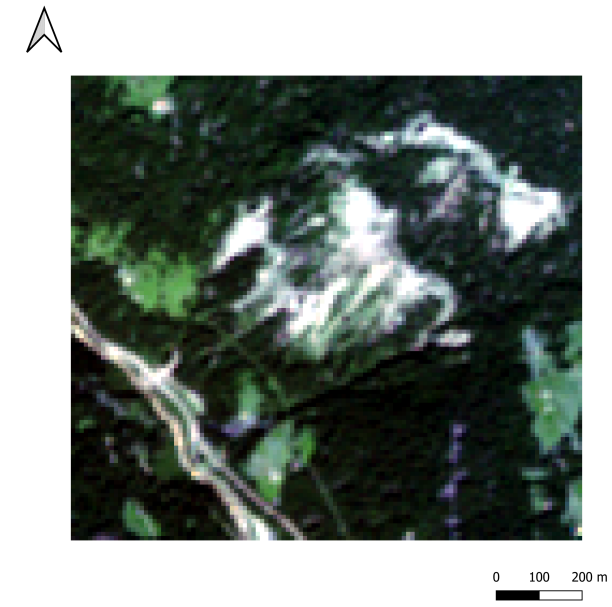


(c) Displacement magnitude output

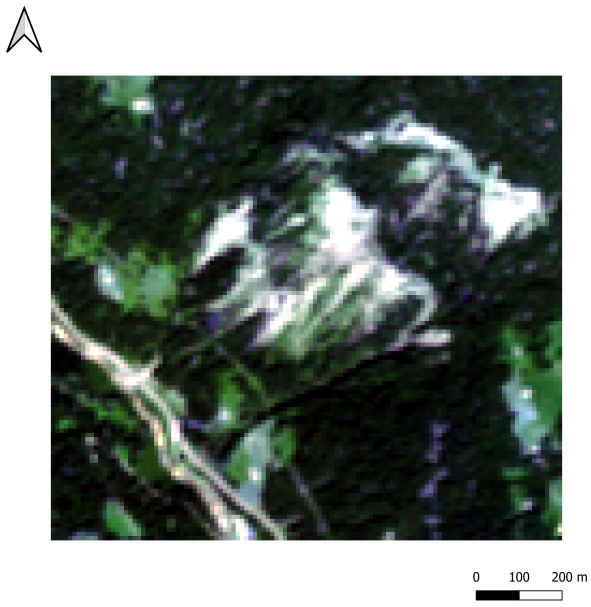


(d) Displacement direction output

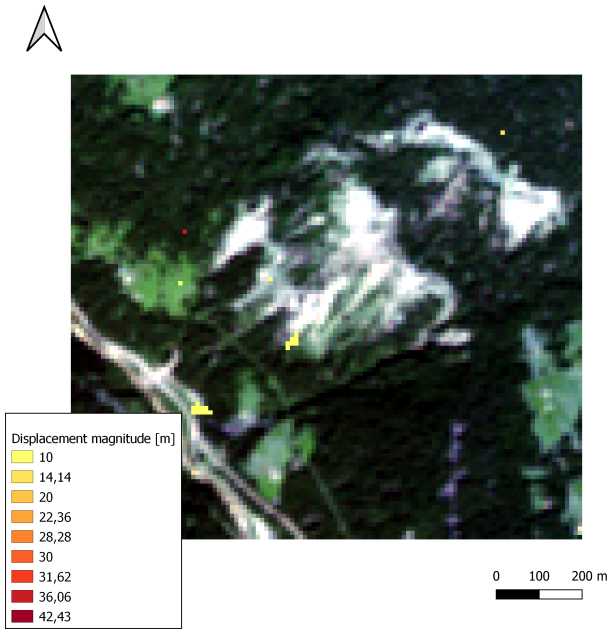
Figure 6.14: 2015-2016 moving master outputs



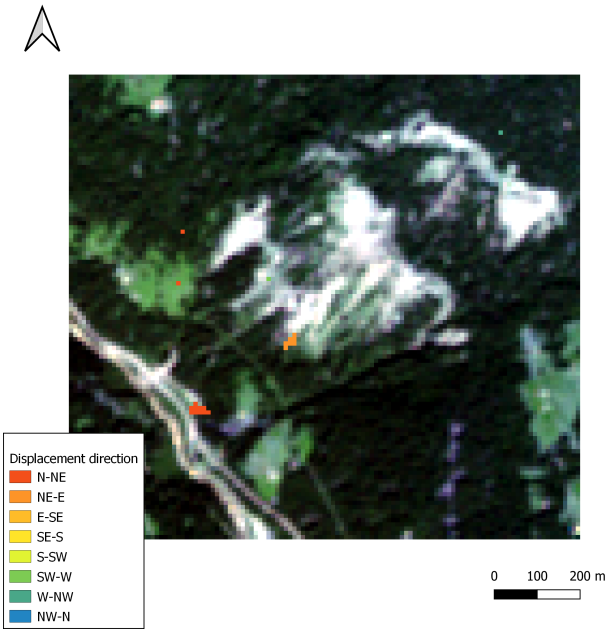
(a) 2016 image *(master)*



(b) 2018 image *(slave)*

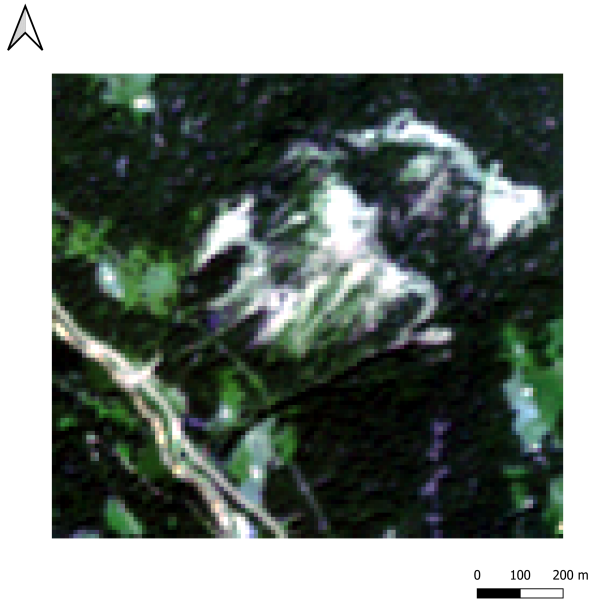


(c) Displacement magnitude output

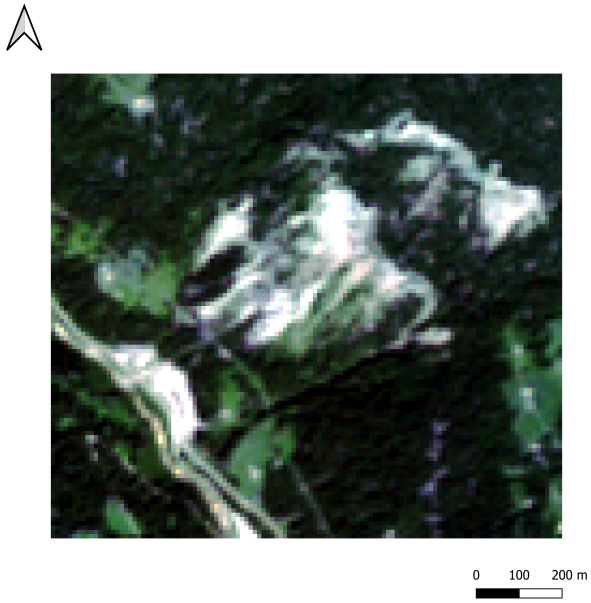


(d) Displacement direction output

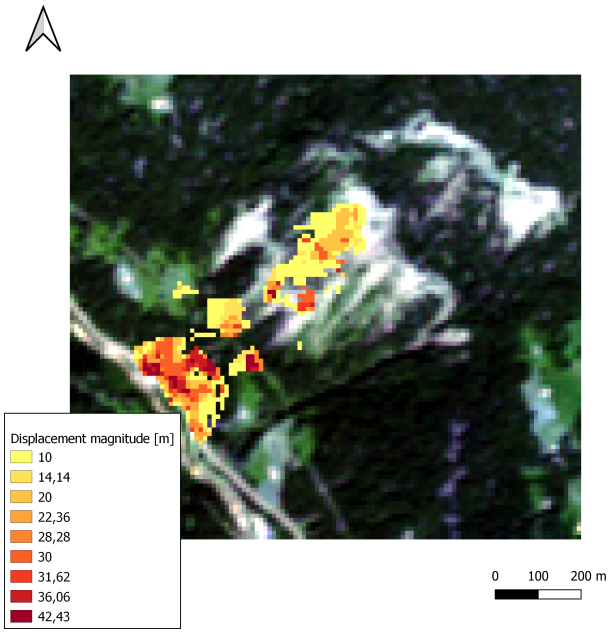
Figure 6.15: 2016-2018 moving master outputs



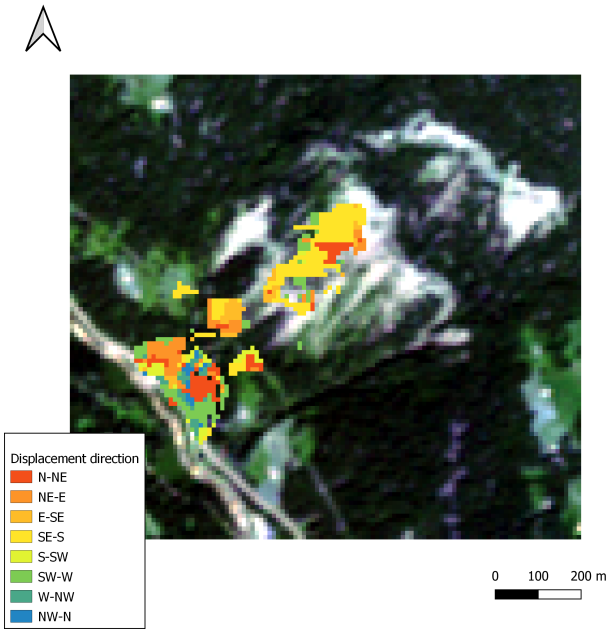
(a) 2018 image *(master)*



(b) 2019 image *(slave)*

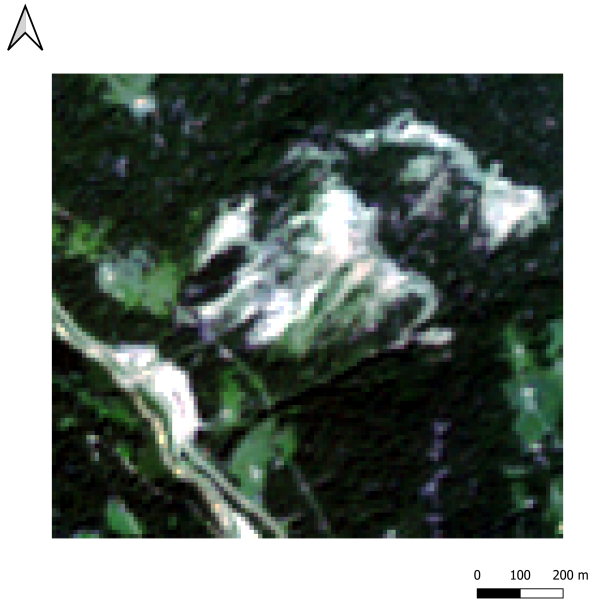


(c) Displacement magnitude output



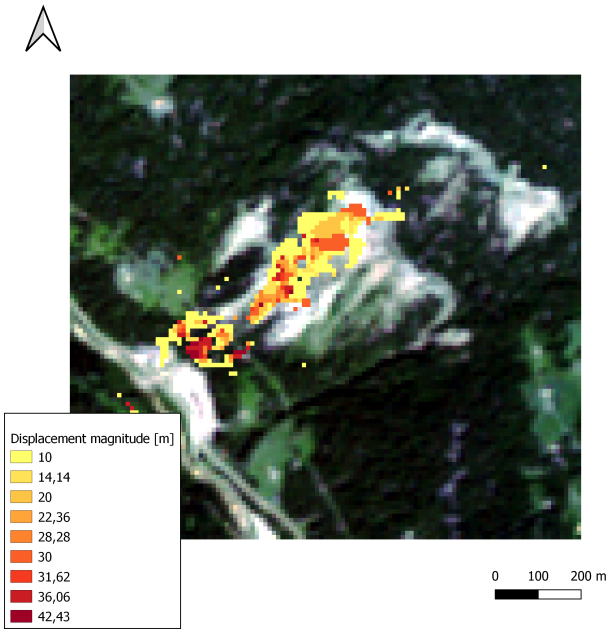
(d) Displacement direction output

Figure 6.16: 2018-2019 moving master outputs

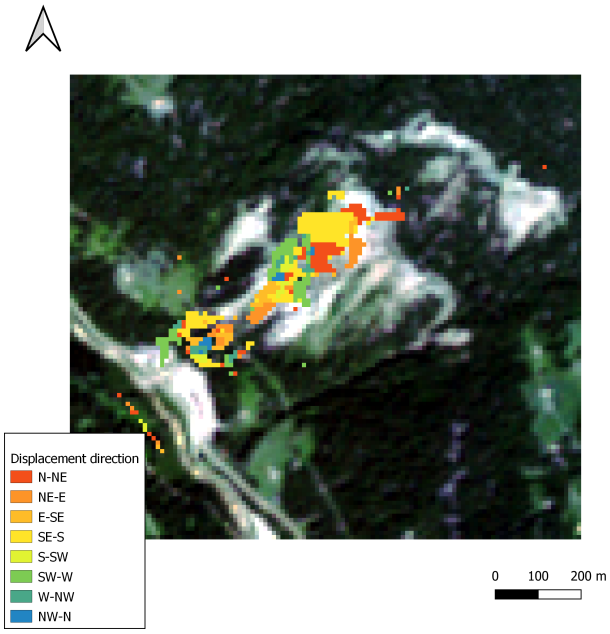


(a) 2019 image *(master)*

(b) 2020 image *(slave)*



(c) Displacement magnitude output



(d) Displacement direction output

Figure 6.17: 2019-2020 moving master outputs

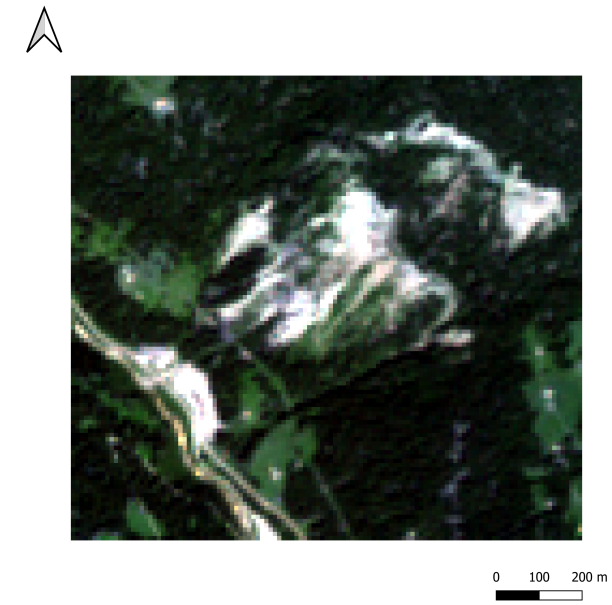
6.2.2 Summer 2019 focus

During the summer of 2019, precisely in August, the Ruinon landslide experienced a very large mass movement in the Western flank of the lower scarp. In order to investigate this episode, three images were selected over the months of July, August and September. In particular, the best image for every month was chosen: Table 6.8 lists all the available images (free from clouds over the landslide and with Relative Orbit number 22), and highlights the selected ones in green.

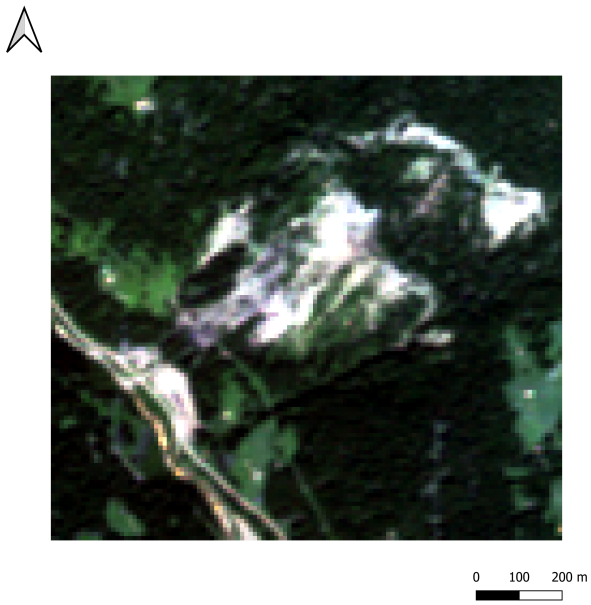
Image code	Sensing date	Selected
S2B_MSIL1C_20190718T101039_N0208_R022_T32TPS	18 July	
S2A_MSIL1C_20190723T101031_N0208_R022_T32TPS	23 July	
S2B_MSIL1C_20190827T101029_N0208_R022_T32TPS	27 August	
S2A_MSIL1C_20190911T101021_N0208_R022_T32TPS	11 September	
S2B_MSIL1C_20190916T101029_N0208_R022_T32TPS	16 September	
S2A_MSIL1C_20190921T101031_N0208_R022_T32TPS	21 September	

Table 6.8: Summer 2019 Sentinel-2 images

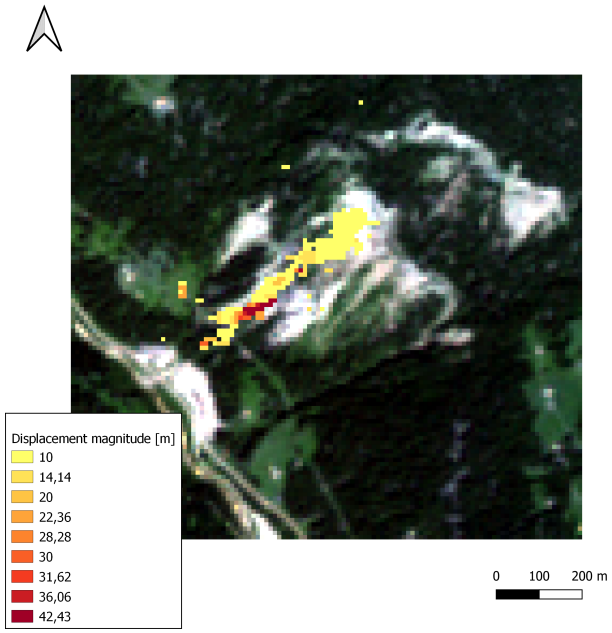
Fixed master Here are reported the results of the displacement monitoring procedure in the summer of 2019 with a *fixed master* approach.



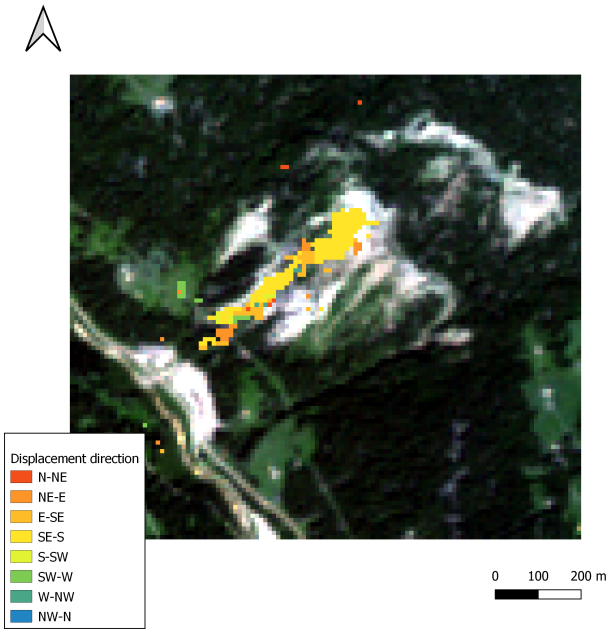
(a) July image *(master)*



(b) August image *(slave)*

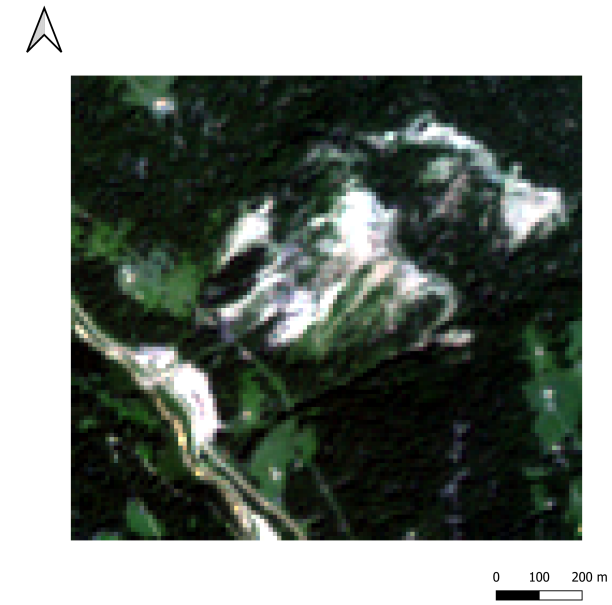


(c) Displacement magnitude output

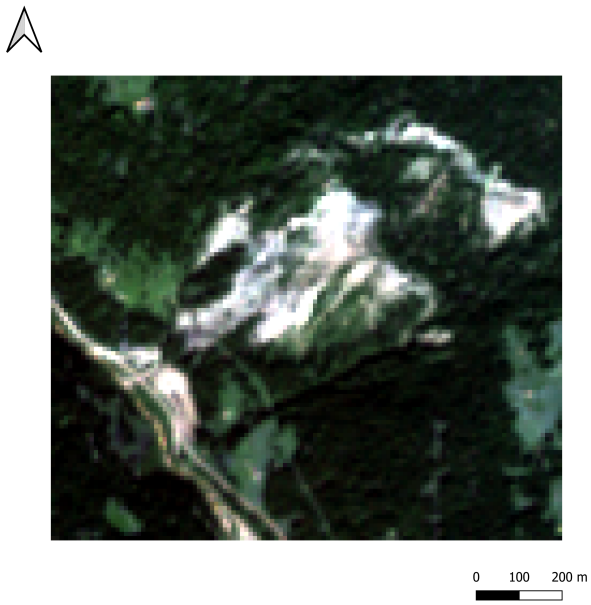


(d) Displacement direction output

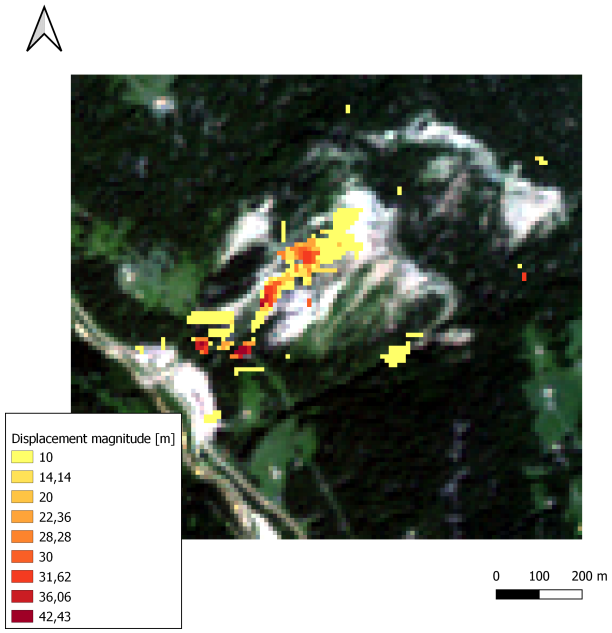
Figure 6.18: July-August fixed master outputs



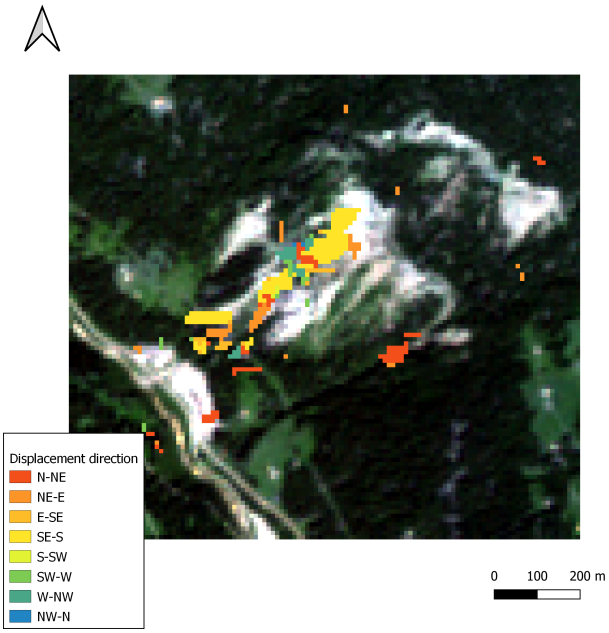
(a) July image *(master)*



(b) September image *(slave)*



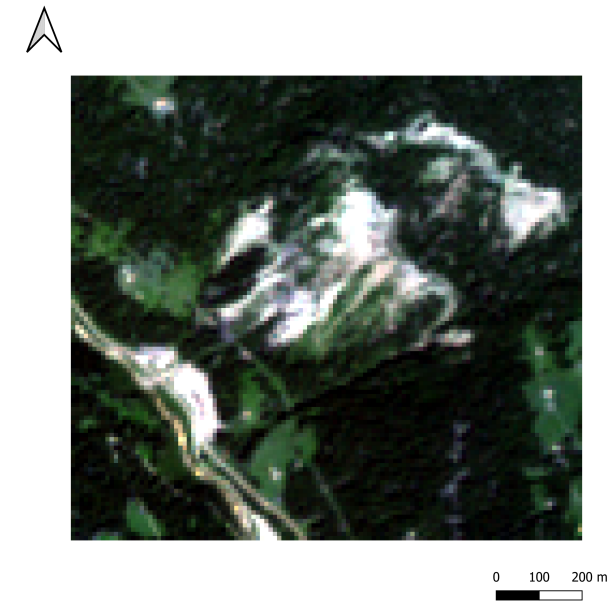
(c) Displacement magnitude output



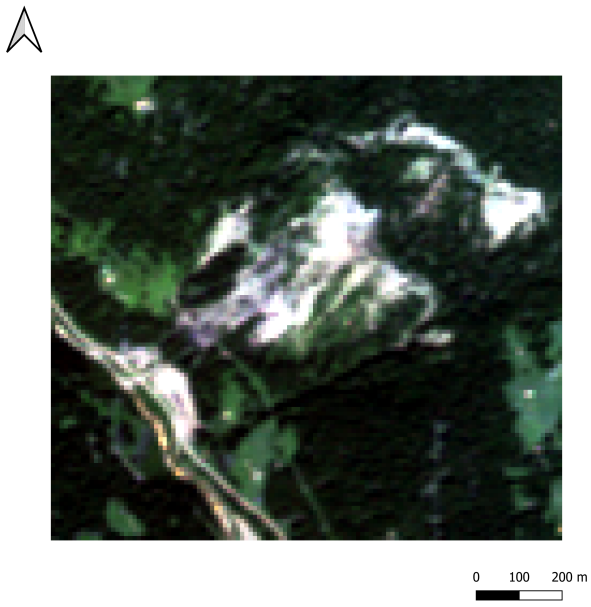
(d) Displacement direction output

Figure 6.19: July-September fixed master outputs

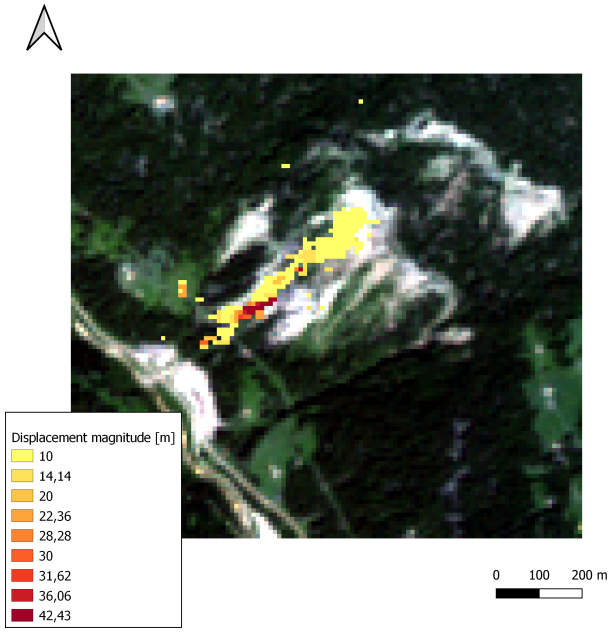
Moving master Lastly, here are reported the results of the displacement monitoring procedure in the summer of 2019 with a *moving master* approach.



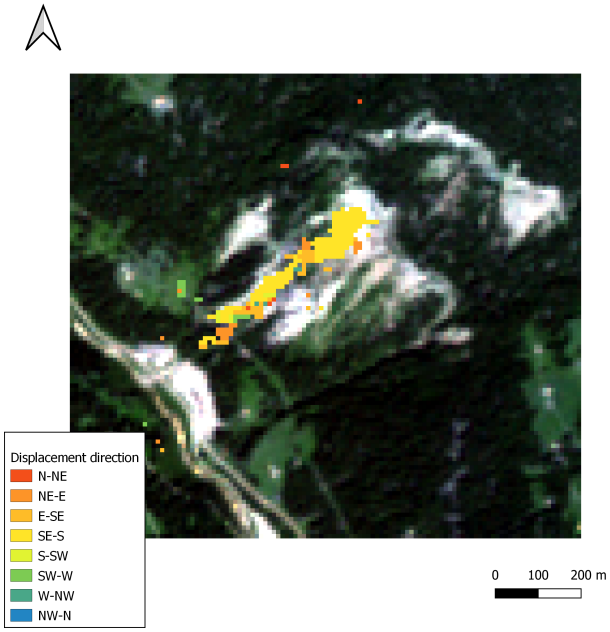
(a) July image *(master)*



(b) August image *(slave)*

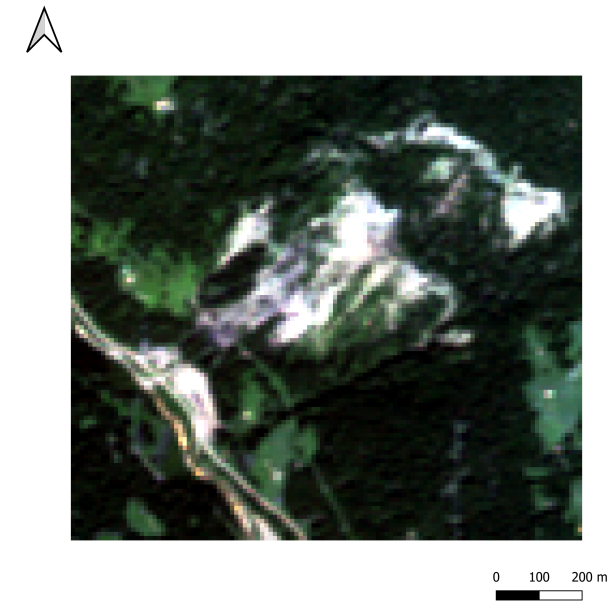


(c) Displacement magnitude output

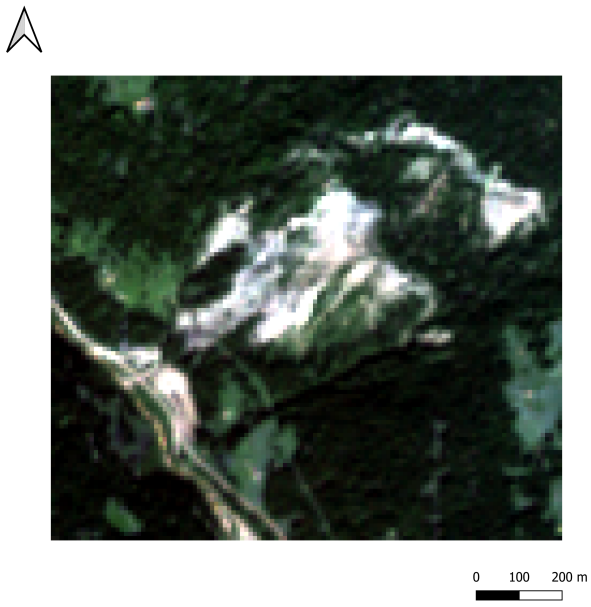


(d) Displacement direction output

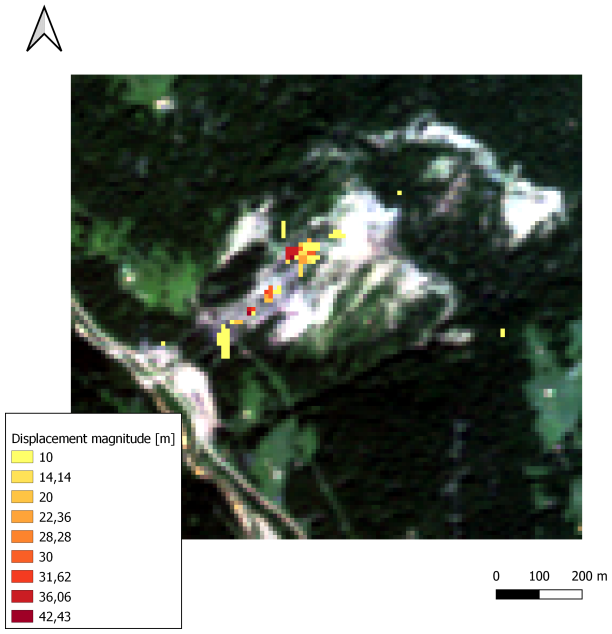
Figure 6.20: July-August moving master outputs



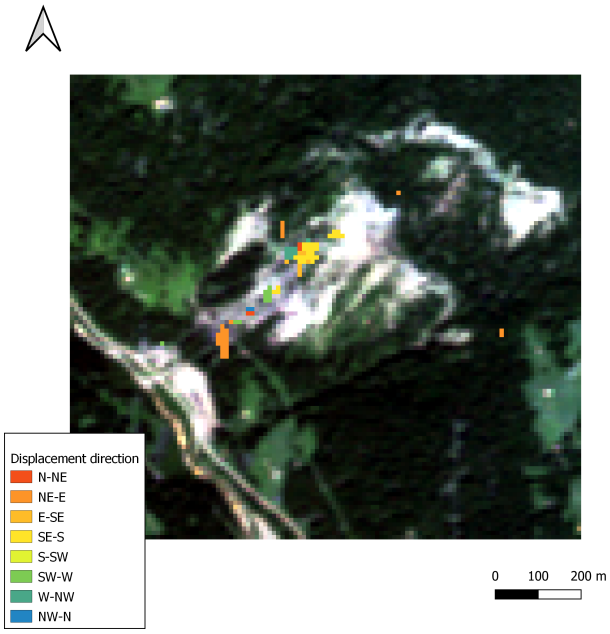
(a) August image *(master)*



(b) September image *(slave)*



(c) Displacement magnitude output



(d) Displacement direction output

Figure 6.21: August-September moving master outputs

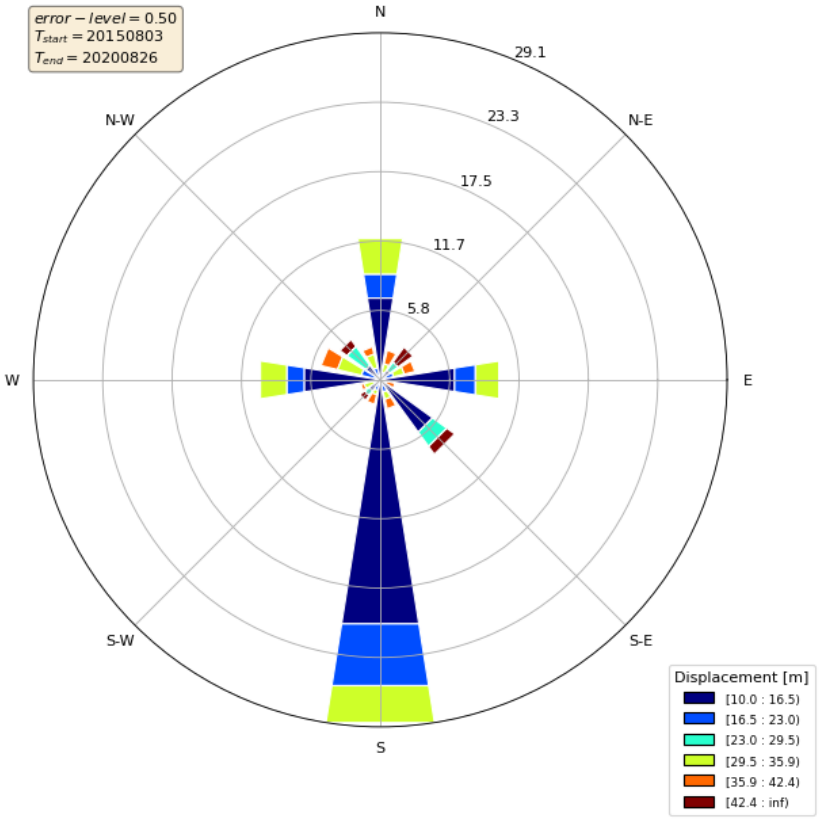
6.2.3 General discussion

As the analysis results show, this procedure correctly identifies the pixel movements occurring between two epochs. On the other hand, it is also important to note the limitations of this analysis: firstly, the smallest displacement that can be identified by cross-correlating two Sentinel-2 images is a displacement of 1 pixel, i.e. a displacement of 10 m. Therefore, smaller movements cannot be sensed by this procedure because of the native resolution of input satellite data. Secondly, errors can arise from the images having differences in terms of co-registration and histogram distribution, since this process highly relies on the images being as aligned and similar as possible. Lastly, skipping a classification process and directly cross-correlating multiband images allows to avoid errors produced by classification, but also exposes the procedure to other problems, such as identifying movements and changes also outside the landslide body.

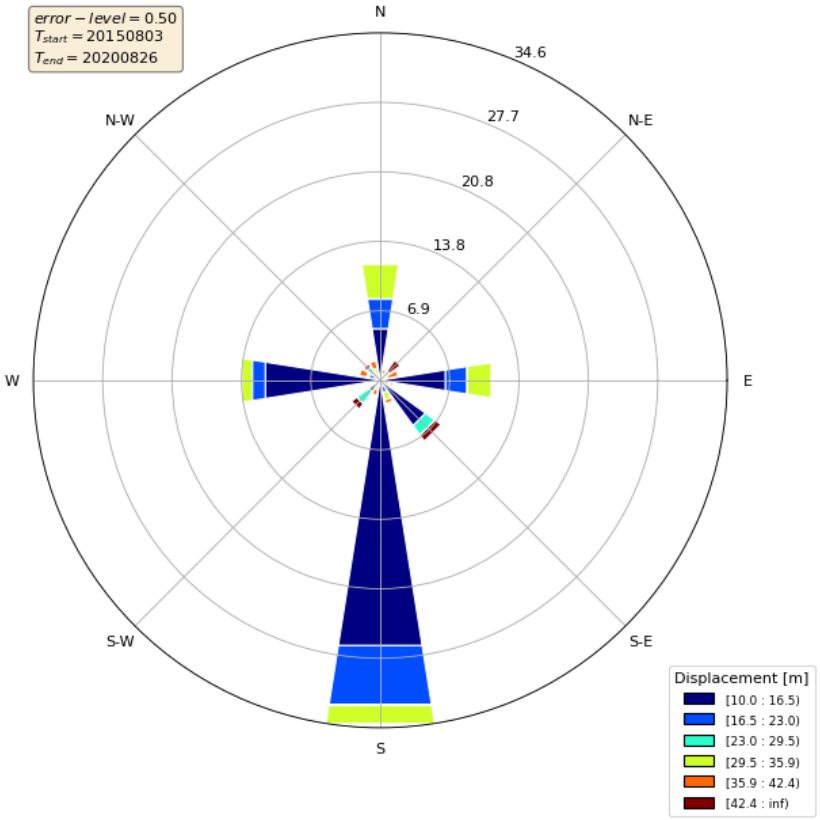
To provide an overview of the obtained results, windrose diagrams were produced (Figure 6.22 and Figure 6.23). A windrose diagram allows to plot the obtained displacements as oriented histograms, therefore giving information about the direction of the movement, the quantity of pixels moving in that direction and the magnitude of those displacements. In addition, the windrose only uses pixels that have a cross-correlation error smaller than an arbitrary threshold, therefore filtering bad results. Even though for all the analyses the error values are rarely above 0.5 (Table 6.9), the error threshold was chosen to be 0.5 in order to include a relevant number of pixels for the diagram computation.

Period	Master strategy	Total pixels	Excluded pixels	Excluded pixels [%]
2015 - 2020	Fixed	2132	101	4.74
	Moving	1362	68	4.99
Summer 2019	Fixed	621	0	0
	Moving	345	0	0

Table 6.9: Excluded pixels due to error value greater than 0.5

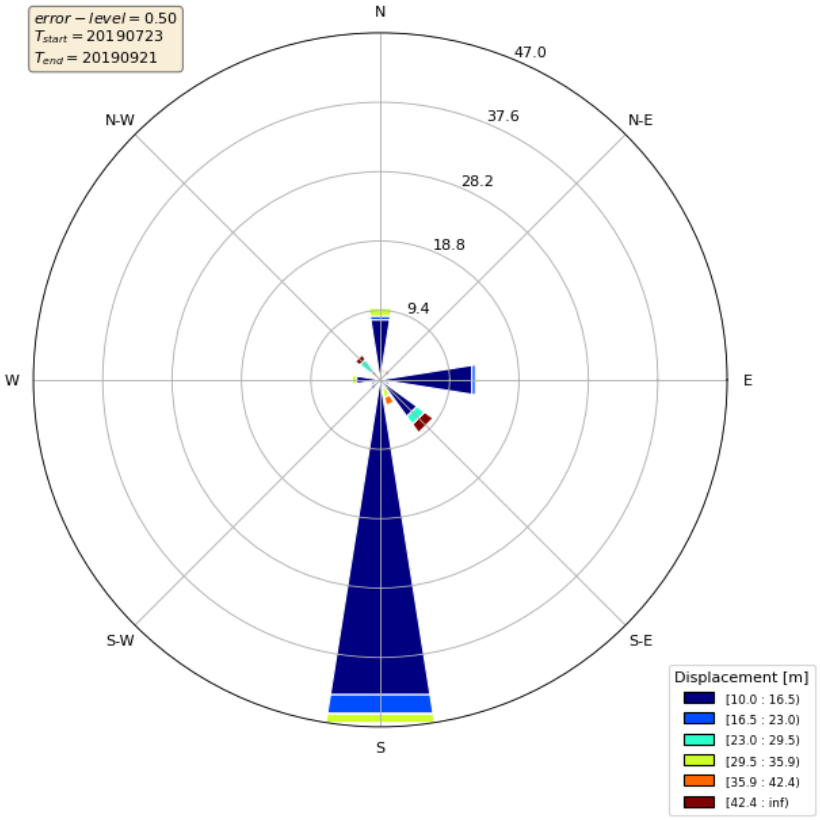


(a) 2015-2020 fixed master windrose diagram

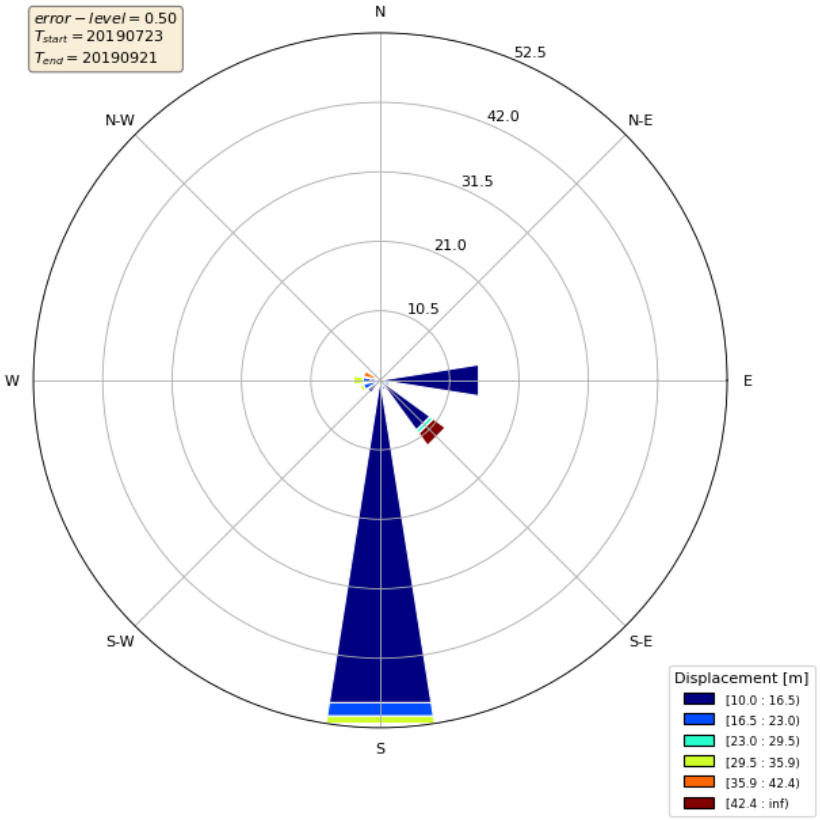


(b) 2015-2020 moving master windrose diagram

Figure 6.22: 2015-2020 windrose diagrams with an error threshold of 0.5



(a) Summer 2019 fixed master windrose diagram



(b) Summer 2019 moving master windrose diagram

Figure 6.23: Summer 2019 windrose diagrams with an error threshold of 0.5

From the windrose diagrams, we can see that the majority of moving pixels is sliding towards South; this is partially consistent with the real movement, even though in reality the Ruinon landslide is mostly moving along the South-West diagonal. Moreover, it seems that there is a high accordance between the diagrams from the two different methods, *fixed* and *moving* master, therefore confirming the validity of the technique.

6.2.4 Validation

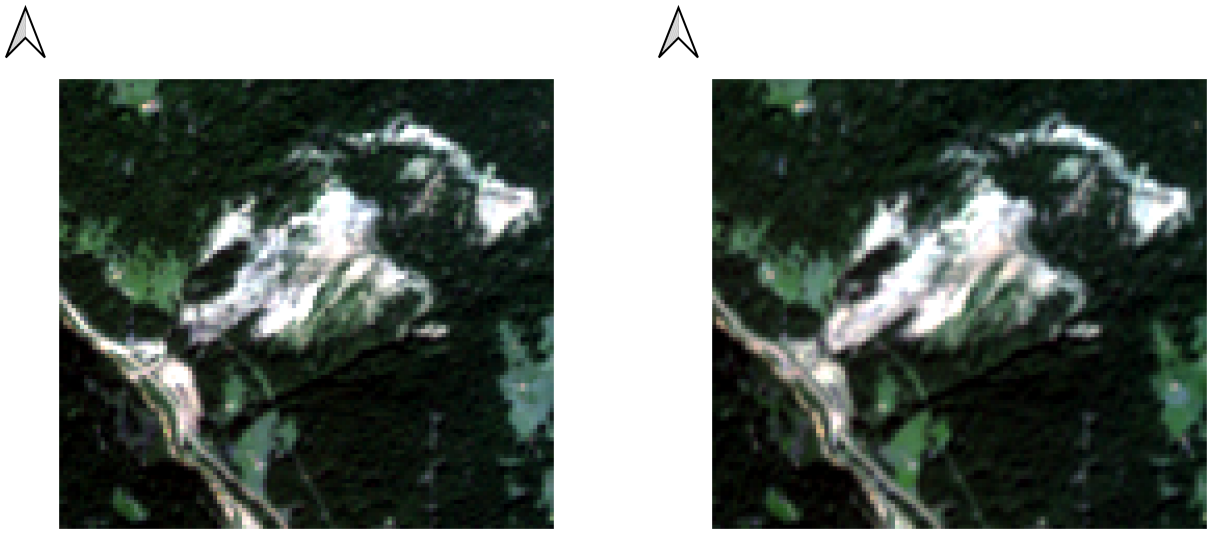
Due to its recent activity, the Ruinon landslide has been monitored with UAV surveys by ARPA Lombardia (the regional agency for environmental protection) and by the GEOLab of Politecnico di Milano. The data collected in these surveys were made available and were used for validating the procedure developed in this work. Table 6.10 shows all the UAV survey dates: in particular the surveys of 27/09/2019 and 10/09/2020 were selected for the validation. The reasons behind this choice were the quality of the survey data and the availability of good satellite images near those dates. The downloaded satellite images were sensed in 21/09/2019 and 15/09/2020.

Survey date	**Survey by**
26/07/2019	ARPA Lombardia
04/09/2019	ARPA Lombardia
20/09/2019	ARPA Lombardia
27/09/2019	ARPA Lombardia
25/10/2019	ARPA Lombardia
10/09/2020	ARPA Lombardia
19/10/2020	ARPA Lombardia
14/07/2021	GEOLab - Politecnico di Milano
31/10/2021	GEOLab - Politecnico di Milano

Table 6.10: UAV surveys of the Ruinon landslide

The displacement monitoring procedure was applied to both the Sentinel-2 images (Figure 6.24) and the RGB images from the surveys resampled to a resolution of 10

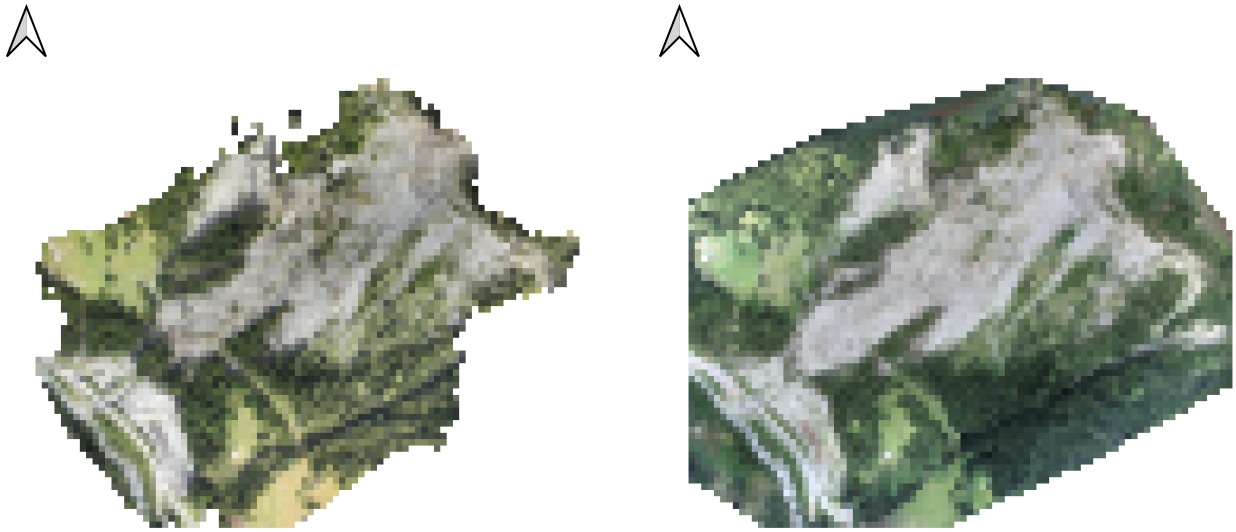
m (Figure 6.25), and the obtained results were compared. Overall the two outputs depict the same portions of the body of the landslide as moving; moreover, also the magnitude of the displacement is similar between the two analyses (Figure 6.26). In particular, the mean value of the differences between the two output rasters is 5 m.



(a) Sentinel-2 image of 21/09/2019

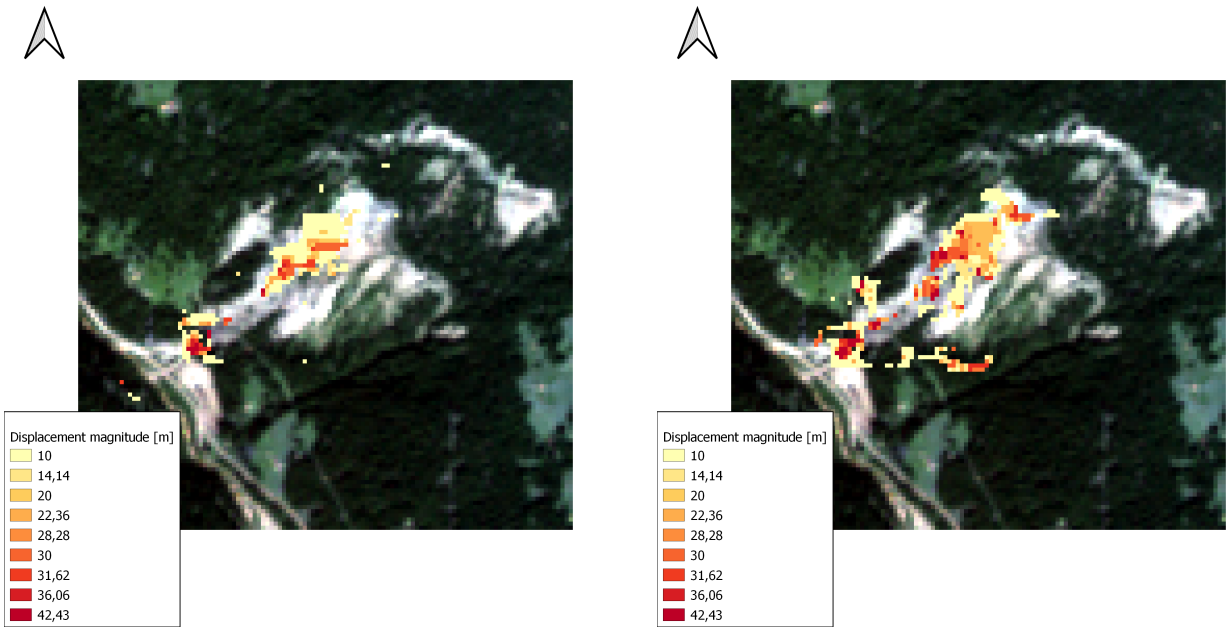
(b) Sentinel-2 image of 15/09/2020

Figure 6.24: Sentinel-2 images used for validation



(a) RGB image from survey of 27/09/2019 (b) RGB image from survey of 10/09/2020

Figure 6.25: RGB images used for validation



(a) Displacement magnitude output from Sentinel-2 images (b) Displacement magnitude output from UAV RGB images

Figure 6.26: Outputs of the two analysis in agreement

In addition, the displacement along the Z axis, that was obtained from the UAV surveys, was inspected looking for confirmations of the movements of the landslide. The Western flank of the body of the landslide was considered, since it corresponds to the part showing a greater movement from the outputs of Sentinel-2. It was seen that between the two surveys an accumulation zone appears in correspondence with the large movement detected by the procedure on the bottom of the Western flank. In particular, a positive increase of the terrain Z of 4 m, due to accumulation of material, was detected by comparing the point clouds from the surveys (Figure 6.27 and Figure 6.28). This was considered in accordance to the movement detected by the developed procedure, thus confirming the validity of the results.

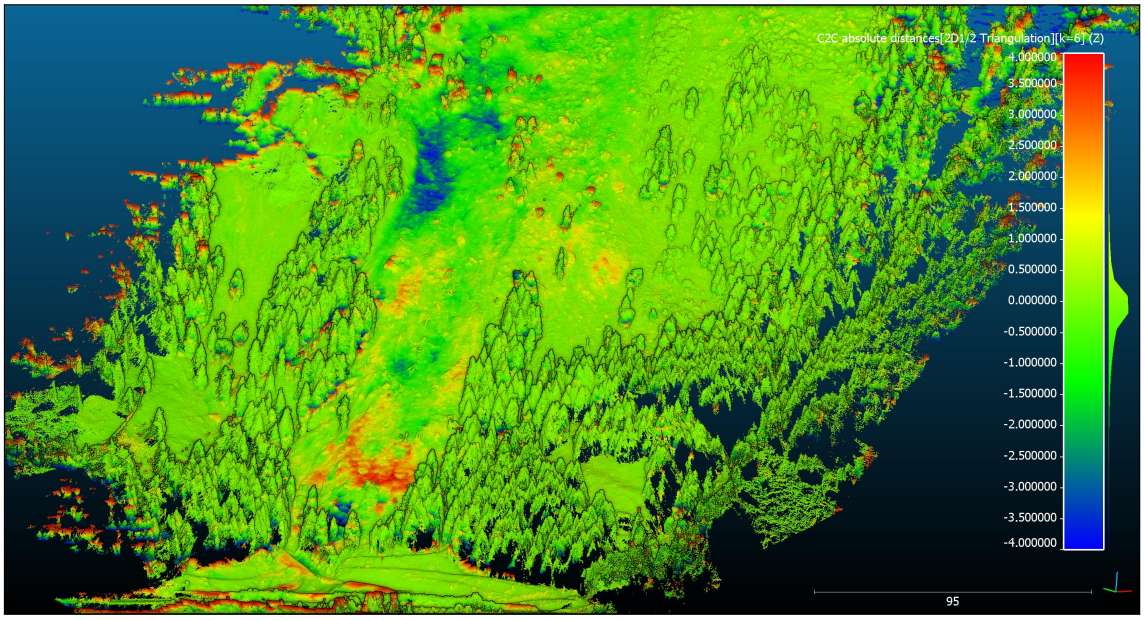


Figure 6.27: Differences along the Z axis between the two UAV surveys

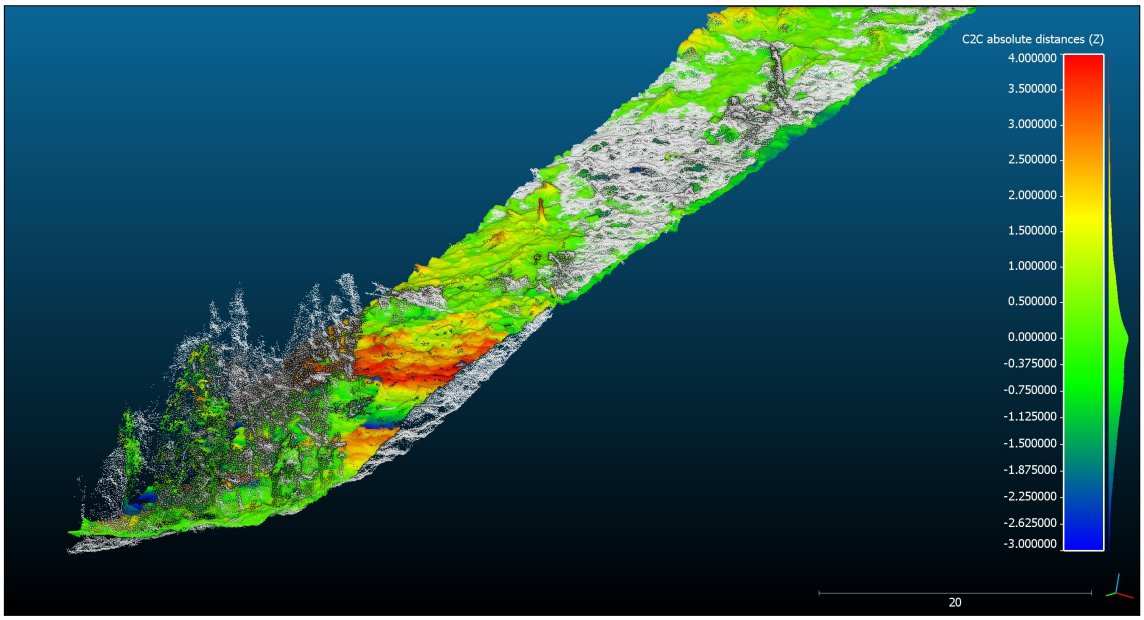


Figure 6.28: Differences along the Z axis between the two UAV surveys: focus on bottom part of the landslide

Chapter 7

Conclusions

7.1 Landslide Susceptibility Mapping

Landslides are one of the most widespread and dangerous geological hazard worldwide (Guzzetti et al., 1999; Reichenbach et al., 2018). Therefore, being able to accurately predict areas that are prone to such phenomena gives the opportunity to control those areas and prevent damages. This makes landslide susceptibility mapping a crucial and fundamental subject.

In particular, the procedure employed in this work was refined with a step by step process with the aim to obtain good and consistent results. In the end, the final procedure led to optimal results for both the areas of interest considered. The reliable performances of the Random Forests technique for landslide susceptibility analysis were confirmed.

Moreover, the inclusion of information about zones that are by hypothesis not prone to landslides, obtained by defining the *NoLS zone*, has proven to strengthen the model and generate valuable results.

Lastly, all the chosen validation metrics agreed when evaluating both the model fitness and the model prediction performances.

In general, the availability of more precise and robust Machine Learning techniques can only benefit landslide susceptibility studies. For the particular case of this procedure, improvements could be made by considering a multitemporal landslide

inventory and including time as a factor in the analysis. Moreover, since the choice of the environmental variables considered in this work was based on the variables that are most used in the literature, exploring the effect of lesser known factors on the model could produce insightful results. Finally, further improvements on the definition of the *NoLS zone* could bring benefits to the analysis.

7.2 Landslide displacement monitoring

The monitoring of active landslides is a matter of primary importance when dealing with mass movement phenomena. The increased availability of high-resolution multitemporal satellite imagery promotes the use of these images for monitoring purposes. While *on the field* monitoring can produce very accurate results, a procedure like the one applied in this work has the advantage to be more flexible, scalable and cheap than an analysis on the field.

In the particular case of this work, an experimental procedure has been developed with the aim of investigating differences between Sentinel-2 satellite images for monitoring the displacement of the Ruinon landslide. Despite being a first stage approach to landslide monitoring, this procedure led to promising results.

Many approaches were considered, varying the main parameters of the procedure (adding or removing a classification phase, considering different intervals between satellite images, modifying the size of the moving window and others). The whole process was progressively improved and refined until it gave satisfactory results.

The core strength of this procedure is that it relies exclusively on FOSS GIS and free and open data. This feature allows the analysis to be easily replicated and empowered.

A procedure like the one adopted in this study could be further improved with the use of higher resolution images. Moreover, mitigating the errors produced by the preprocessing phases, such as the image co-registration process, could enhance the precision and the robustness of this technique.

Acknowledgements

The author would like to thank ARPA Lombardia, specifically Doctors Dario Bellingeri, Luca Dei Cas, Alessandro Menin and Enrico Zini, for supporting this research with data and suggestions.

Bibliography

Aditian, A., Kubota, T., and Shinohara, Y. (2018). Comparison of GIS-based landslide susceptibility models using frequency ratio, logistic regression, and artificial neural network in a tertiary region of Ambon, Indonesia. *Geomorphology*, 318:101–111.

ArcGIS documentation (2021). Shapefiles. https://doc.arcgis.com/en/arcgis-online/reference/shapefiles.htm.

Ayalew, L. and Yamagishi, H. (2005). The application of GIS-based logistic regression for landslide susceptibility mapping in the Kakuda-Yahiko Mountains, Central Japan. *Geomorphology*, 65(1-2):15–31.

Berthier, E., Vadon, H., Baratoux, D., Arnaud, Y., Vincent, C., Feigl, K., Rémy, F., and Legrésy, B. (2005). Surface motion of mountain glaciers derived from satellite optical imagery. *Remote Sensing of Environment*, 95(1):14–28.

Breiman, L. (2001). Random forests. *Machine Learning*, 45(1):5–32.

Brovelli, M., Minghini, M., Moreno-Sanchez, R., and Oliveira, R. (2017). Free and open source software for geospatial applications (FOSS4G) to support Future Earth. *International Journal of Digital Earth*, 10(4):386–404.

Carlà, T., Gigli, G., Lombardi, L., Nocentini, M., and Casagli, N. (2021). Monitoring and analysis of the exceptional displacements affecting debris at the top of a highly disaggregated rockslide. *Engineering Geology*, 294.

Carrara, A. (1983). Multivariate models for landslide hazard evaluation. *Journal of the International Association for Mathematical Geology*, 15(3):403–426.

Catani, F., Lagomarsino, D., Segoni, S., and Tofani, V. (2013). Landslide susceptibility estimation by random forests technique: Sensitivity and scaling issues. *Natural Hazards and Earth System Sciences*, 13(11):2815–2831.

Chung, C.-J. F. and Fabbri, A. G. (1999). Probabilistic prediction models for landslide hazard mapping. *Photogrammetric Engineering and Remote Sensing*, 65:1389–1400.

Chung, C. J. F. and Fabbri, A. G. (2003). Validation of spatial prediction models for landslide hazard mapping. *Natural Hazards*, 30(3):451–472.

Colesanti, C. and Wasowski, J. (2006). Investigating landslides with space-borne synthetic aperture radar (sar) interferometry. *Engineering Geology*, 88(3):173–199.

Copernicus Programme (2021). Copernicus Open Access Hub. https://scihub.copernicus.eu.

Coppin, P. and Bauer, M. (1996). Digital change detection in forest ecosystems with remote sensing imagery. *Remote Sensing Reviews*, 13(3-4):207–234. cited By 456.

Crocker, R., Matthews, D., Emery, W., and Baldwin, D. (2007). Computing coastal ocean surface currents from infrared and ocean color satellite imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 45(2):435–447. cited By 71.

Cruden, D. and Varnes, D. (1996). Landslide types and processes. *Special Report - National Research Council, Transportation Research Board*, 247:36–75.

Dewan, A. and Yamaguchi, Y. (2009). Land use and land cover change in greater dhaka, bangladesh: Using remote sensing to promote sustainable urbanization. *Applied Geography*, 29(3):390–401. cited By 577.

Dou, J., Yunus, A. P., Tien Bui, D., Merghadi, A., Sahana, M., Zhu, Z., Chen, C. W., Khosravi, K., Yang, Y., and Pham, B. T. (2019). Assessment of advanced random forest and decision tree algorithms for modeling rainfall-induced landslide susceptibility in the Izu-Oshima Volcanic Island, Japan. *Science of the Total Environment*, 662:332–346.

Emami, S., Yousefi, S., Pourghasemi, H., Tavangar, S., and Santosh, M. (2020). A comparative study on machine learning modeling for mass movement susceptibility mapping (a case study of Iran). *Bulletin of Engineering Geology and the Environment*, 79(10):5291–5308.

Fang, Z., Wang, Y., Peng, L., and Hong, H. (2020). Integration of convolutional neural network and conventional machine learning classifiers for landslide susceptibility mapping. *Computers and Geosciences*, 139.

Fang, Z., Wang, Y., Peng, L., and Hong, H. (2021). A comparative study of heterogeneous ensemble-learning techniques for landslide susceptibility mapping. *International Journal of Geographical Information Science*, 35(2):321–347.

Freeman, E. A., Frescino, T. S., and Moisen, G. G. (2016). ModelMap : an R Package for Model Creation and Map Production. pages 1–43.

Fulkerson, B., Michie, D., Spiegelhalter, D. J., and Taylor, C. C. (1995). Machine Learning, Neural and Statistical Classification. *Technometrics*, 37(4):459.

Goetz, J. N., Brenning, A., Petschko, H., and Leopold, P. (2015). Evaluating machine learning and statistical prediction techniques for landslide susceptibility modeling. *Computers and Geosciences*, 81:1–11.

Google (2021). Google Earth Engine. https://earthengine.google.com.

Guzzetti, F., Carrara, A., Cardinali, M., and Reichenbach, P. (1999). Landslide hazard evaluation: a review of current techniques andtheir application in a multi-scale study, Central Italy. *Geomorphology*, 13(6):1995.

Guzzetti, F., Mondini, A. C., Cardinali, M., Fiorucci, F., Santangelo, M., and Chang, K. T. (2012). Landslide inventory maps: New tools for an old problem. *Earth-Science Reviews*, 112(1-2):42–66.

Guzzetti, F., Reichenbach, P., Ardizzone, F., Cardinali, M., and Galli, M. (2006). Estimating the quality of landslide susceptibility models. *Geomorphology*, 81(1-2):166–184.

Highland, L. M. and Bobrowsky, P. (2008). The landslide Handbook - A guide to understanding landslides. *US Geological Survey Circular*, (1325):1–147.

Infrastructure for Spatial Information in Europe (2019). INSPIRE registry, Coordinate reference systems. https://inspire.ec.europa.eu/theme/rs.

Kennedy, R., Yang, Z., and Cohen, W. (2010). Detecting trends in forest disturbance and recovery using yearly landsat time series: 1. landtrendr - temporal segmentation algorithms. *Remote Sensing of Environment*, 114(12):2897–2910. cited By 815.

Lee, S., Ryu, J. H., Lee, M. J., and Won, J. S. (2003). Use of an artificial neural network for analysis of the susceptibility to landslides at Boun, Korea. *Environmental Geology*, 44(7):820–833.

Lee, S. and Sambath, T. (2006). Landslide susceptibility mapping in the Damrei Romel area, Cambodia using frequency ratio and logistic regression models. *Environmental Geology*, 50(6):847–855.

Leese, J. A., Novak, C. S., and Clark, B. B. (1971). Automated technique for obtaining cloud motion from geosynchronous satellite data using cross correlation. *Journal of Applied Meteorology*, 10(1):118–132. cited By 228.

Liaw, A. and Wiener, M. (2002). Classification and Regression by randomForest. *R News*, 2(3):18–22.

Longoni, L., Papini, M., Brambilla, D., Barazzetti, L., Roncoroni, F., Scaioni, M., and Ivanov, V. I. (2016). Monitoring riverbank erosion in mountain catchments using terrestrial laser scanning. *Remote Sensing*, 8(3).

Ma, Z., Mei, G., and Piccialli, F. (2020). Machine learning for landslides prevention: a survey. *Neural Computing and Applications*, 8.

Machichi, M., Saadane, A., and Guth, P. (2020). On the viability of Neural Networks for landslide susceptibility mapping in the Rif, North of Morocco. In *Proceedings - 2020 IEEE International Conference of Moroccan Geomatics, MORGEO 2020*.

Malczewski, J. (1999). *GIS and multicriteria decision analysis*. John Wiley & Sons.

Neuland, H. (1976). A prediction model of landslips. *CATENA*, 3(2):215–230.

Ninnis, R. M., Emery, W. J., and Collins, M. J. (1986). Automated extraction of pack ice motion from advanced very high resolution radiometer imagery. *Journal of Geophysical Research: Oceans*, 91(C9):10725–10734.

OSGeo (2021). GRASS GIS. https://grass.osgeo.org/.

Oxoli, D., Brovelli, M. A., Frizzi, D., and Martinati, S. (2020). Detection of land cover displacements through time-series analysis of multispectral satellite imagery: Application to desert. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B3-2020:739–744.

Pham, B. T., Pradhan, B., Tien Bui, D., Prakash, I., and Dholakia, M. B. (2016). A comparative study of different machine learning methods for landslide susceptibility assessment: A case study of Uttarakhand area (India). *Environmental Modelling and Software*, 84:240–250.

Pourghasemi, H. R. and Rahmati, O. (2018). Prediction of the landslide susceptibility: Which algorithm, which precision? *Catena*, 162(December 2017):177–192.

Prakash, N., Manconi, A., and Loew, S. (2020). Mapping landslides on EO data: Performance of deep learning models vs. Traditional machine learning models. *Remote Sensing*, 12(3).

Python software foundation (2021). Python. https://www.python.org/.

QGIS (2021). QGIS - A Free and Open Source Geographic Information System. https://www.qgis.org/en/site/.

R foundation (2021). R-project. https://www.r-project.org/.

Reichenbach, P., Rossi, M., Malamud, B. D., Mihir, M., and Guzzetti, F. (2018). A review of statistically-based landslide susceptibility models. *Earth-Science Reviews*, 180(November 2017):60–91.

Rossi, M., Guzzetti, F., Reichenbach, P., Mondini, A. C., and Peruccacci, S. (2010). Optimal landslide susceptibility zonation based on multiple forecasts. *Geomorphology*, 114(3):129–142.

RStudio (2021). RStudio Workbench. https://www.rstudio.com/.

Saito, H., Nakayama, D., and Matsuyama, H. (2009). Comparison of landslide susceptibility based on a decision-tree model and actual landslide occurrence: The Akaishi Mountains, Japan. *Geomorphology*, 109(3-4):108–121.

Saito, T. and Rehmsmeier, M. (2015). The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. *PLoS ONE*, 10(3):1–21.

Thi Ngo, P. T., Panahi, M., Khosravi, K., Ghorbanzadeh, O., Kariminejad, N., Cerda, A., and Lee, S. (2021). Evaluation of deep learning algorithms for national scale landslide susceptibility mapping of Iran. *Geoscience Frontiers*, 12(2):505–519.

Tien Bui, D., Tuan, T. A., Klempe, H., Pradhan, B., and Revhaug, I. (2016). Spatial prediction models for shallow landslide hazards: a comparative assessment of the efficacy of support vector machines, artificial neural networks, kernel logistic regression, and logistic model tree. *Landslides*, 13(2):361–378.

Wang, Y., Fang, Z., and Hong, H. (2019). Comparison of convolutional neural networks for landslide susceptibility mapping in Yanshan County, China. *Science of the Total Environment*, 666:975–993.

Yao, X., Tham, L. G., and Dai, F. C. (2008). Landslide susceptibility mapping based on Support Vector Machine: A case study on natural slopes of Hong Kong, China. *Geomorphology*, 101(4):572–582.

Yordanov, V., Biagi, L., Truong, X. Q., Tran, V. A., and Brovelli, M. A. (2021). an Overview of Geoinformatics State-of-the-Art Techniques for Landslide Monitoring and Mapping. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W2-(October):205–212.

Yordanov, V. and Brovelli, M. A. (2020a). Application of various strategies and methodologies for landslide susceptibility maps on a basin scale: the case study of Val Tartano, Italy. *Applied Geomatics*.

Yordanov, V. and Brovelli, M. A. (2020b). Comparing model performance metrics for landslide susceptibility mapping. *International Archives of the*

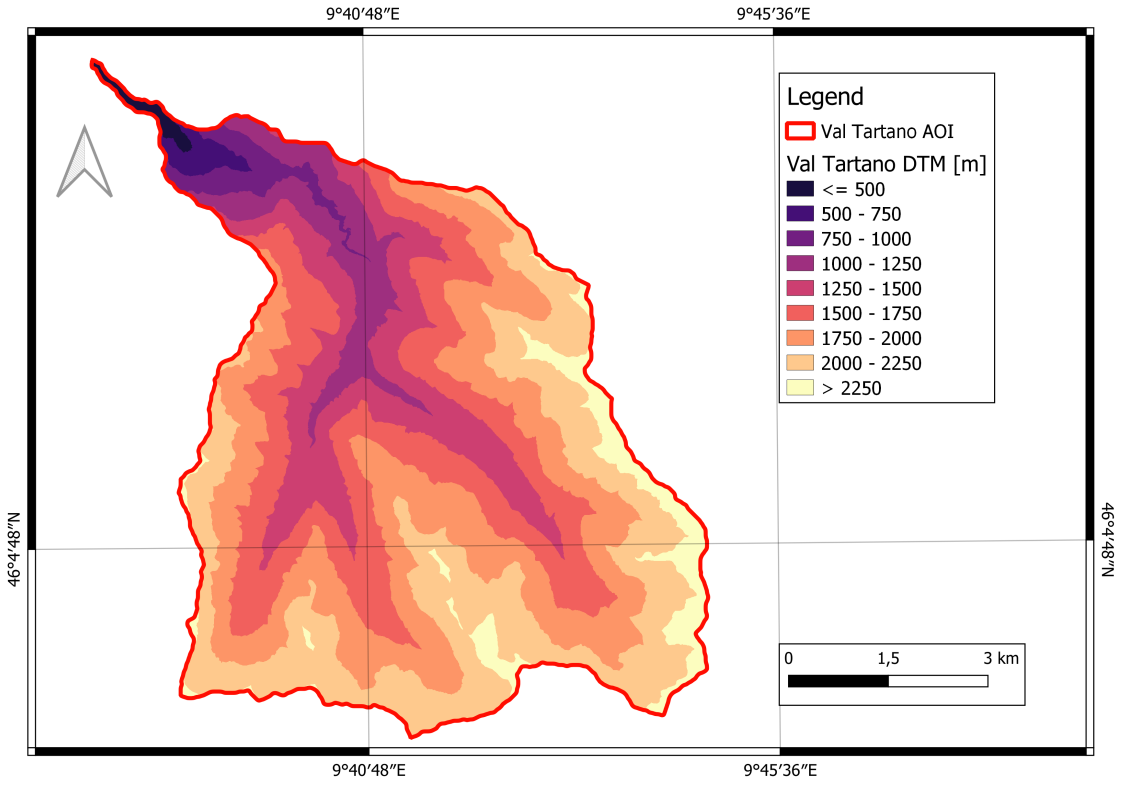
Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives, 43(B3):1277–1284.

You, M., Filippi, A. M., Güneralp, I., and Güneralp, B. (2017). What is the direction of land change? A new approach to land-change analysis. *Remote Sensing*, 9(8):1–18.

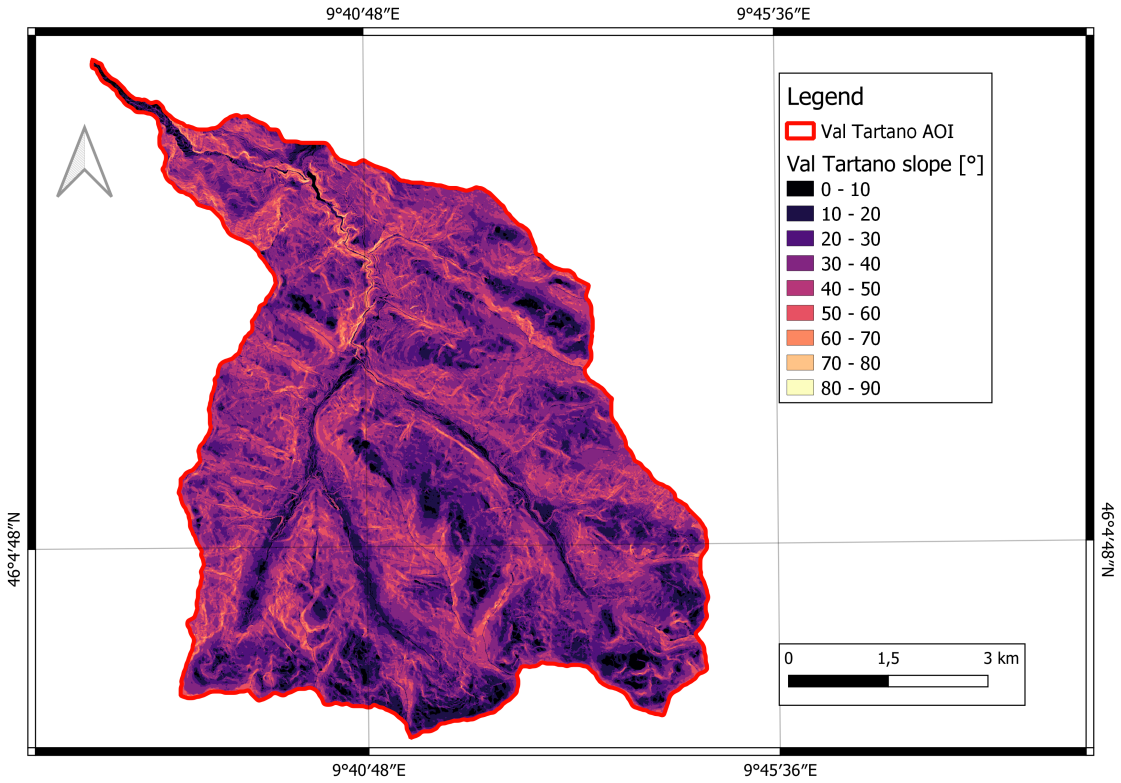
Zhou, C., Yin, K., Cao, Y., Ahmed, B., Li, Y., Catani, F., and Pourghasemi, H. (2018). Landslide susceptibility modeling applying machine learning methods: A case study from Longju in the Three Gorges Reservoir area, China. *Computers and Geosciences*, 112:23–37.

Appendix A

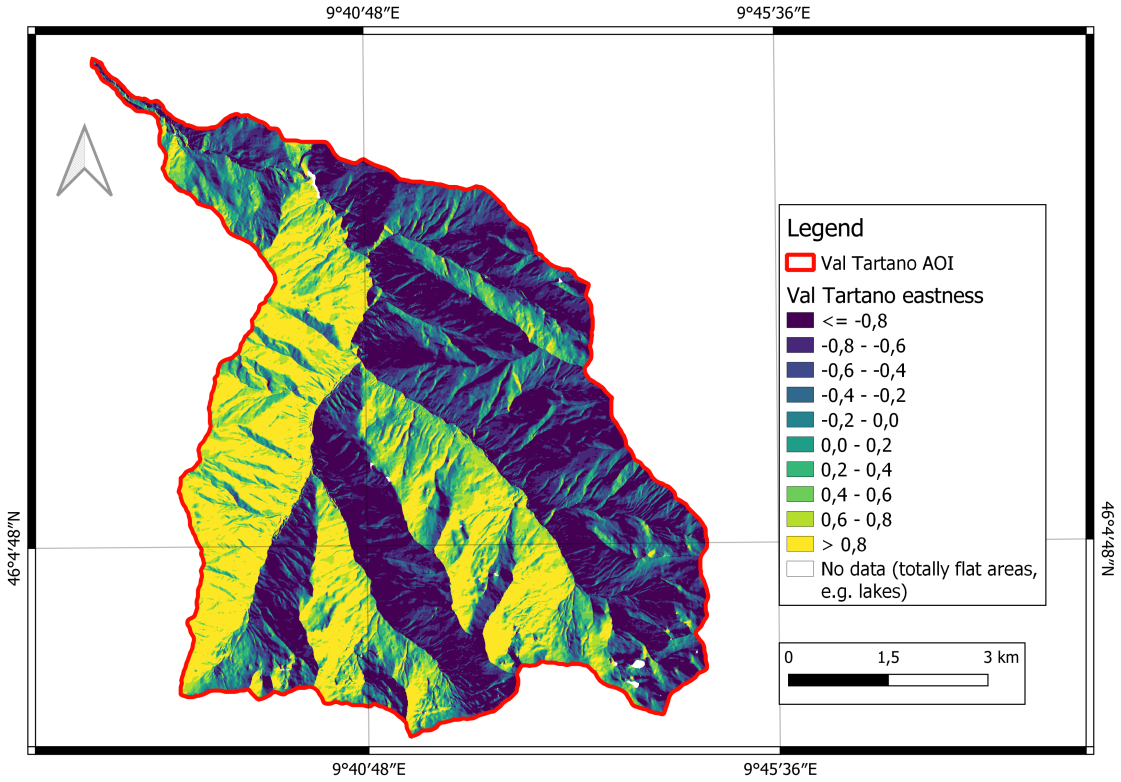
Val Tartano terrain variables



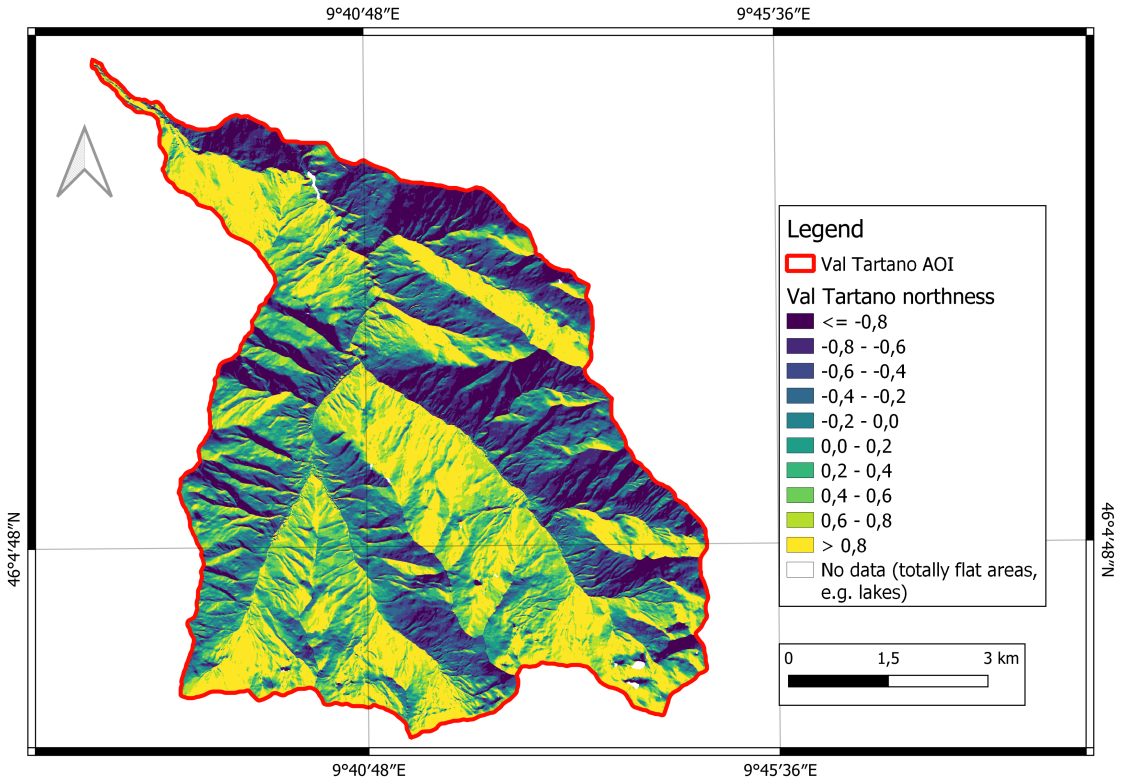
(a) Val Tartano DTM



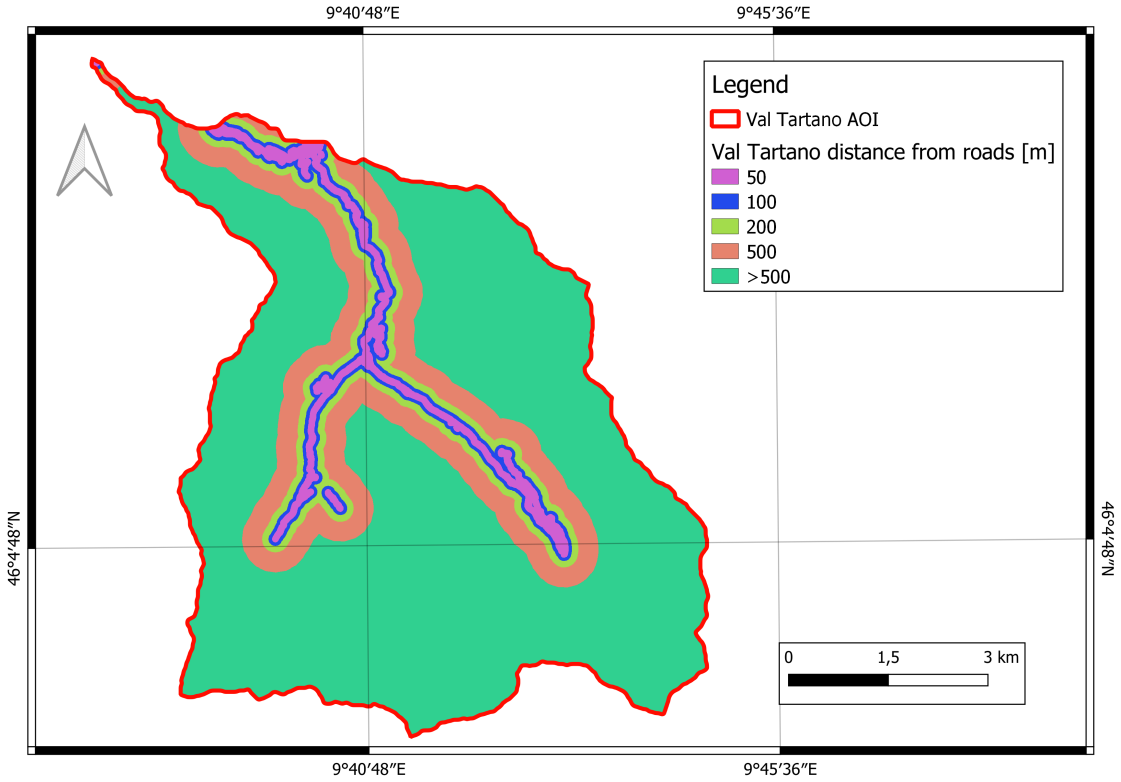
(a) Val Tartano slope



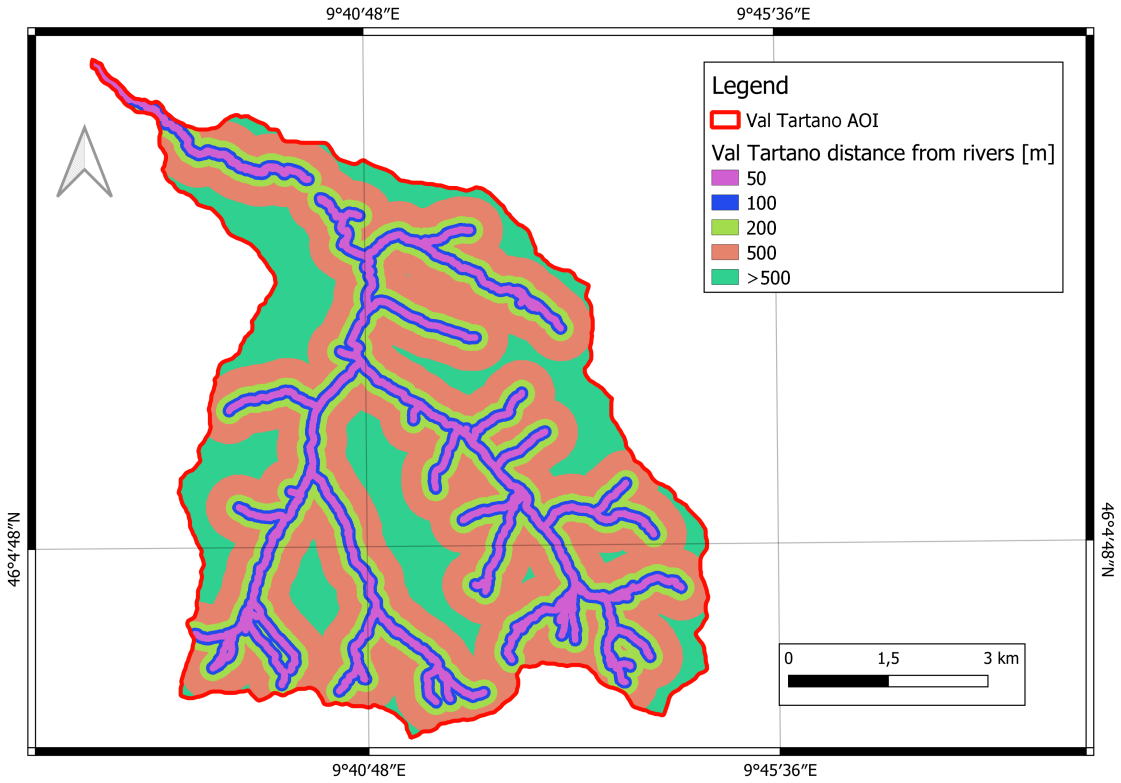
(b) Val Tartano eastness



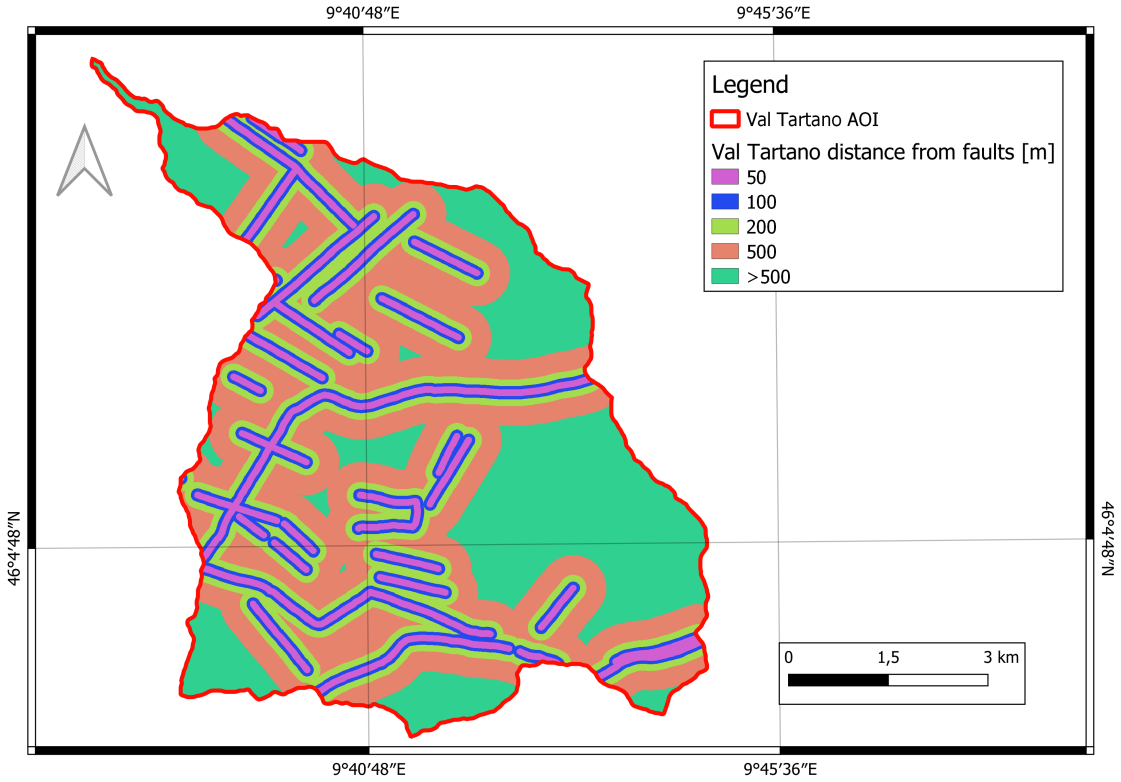
(c) Val Tartano northness



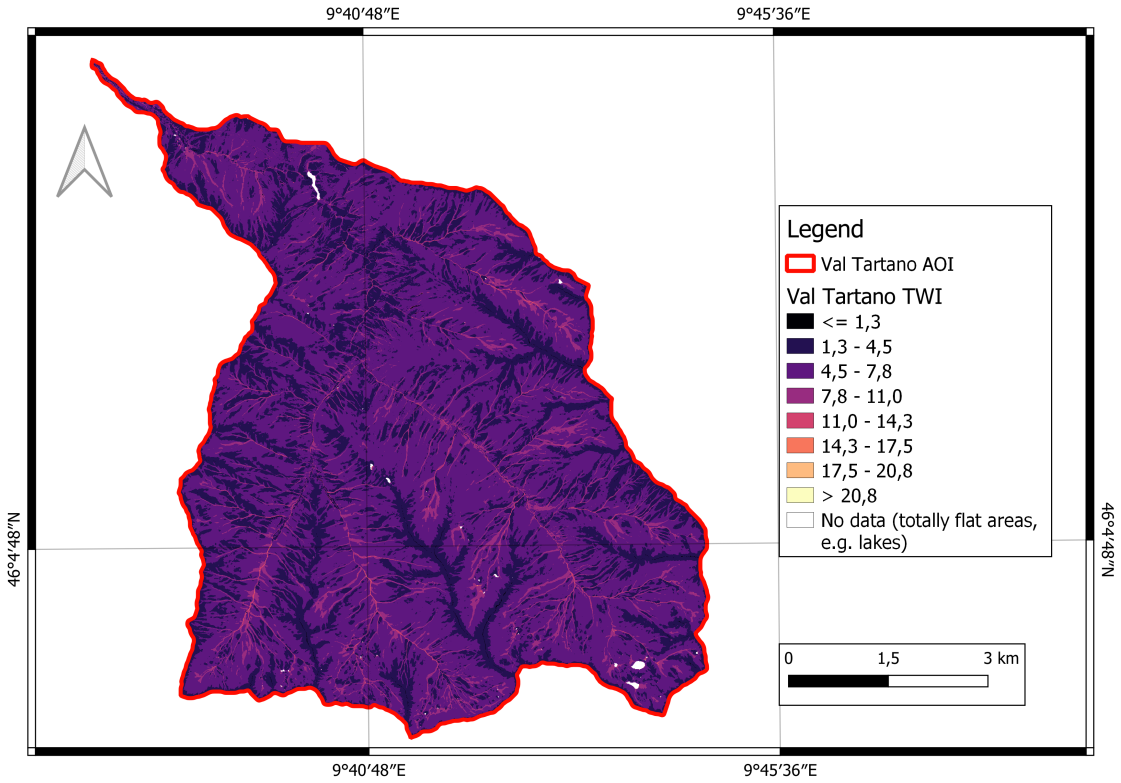
(d) Val Tartano distance from roads



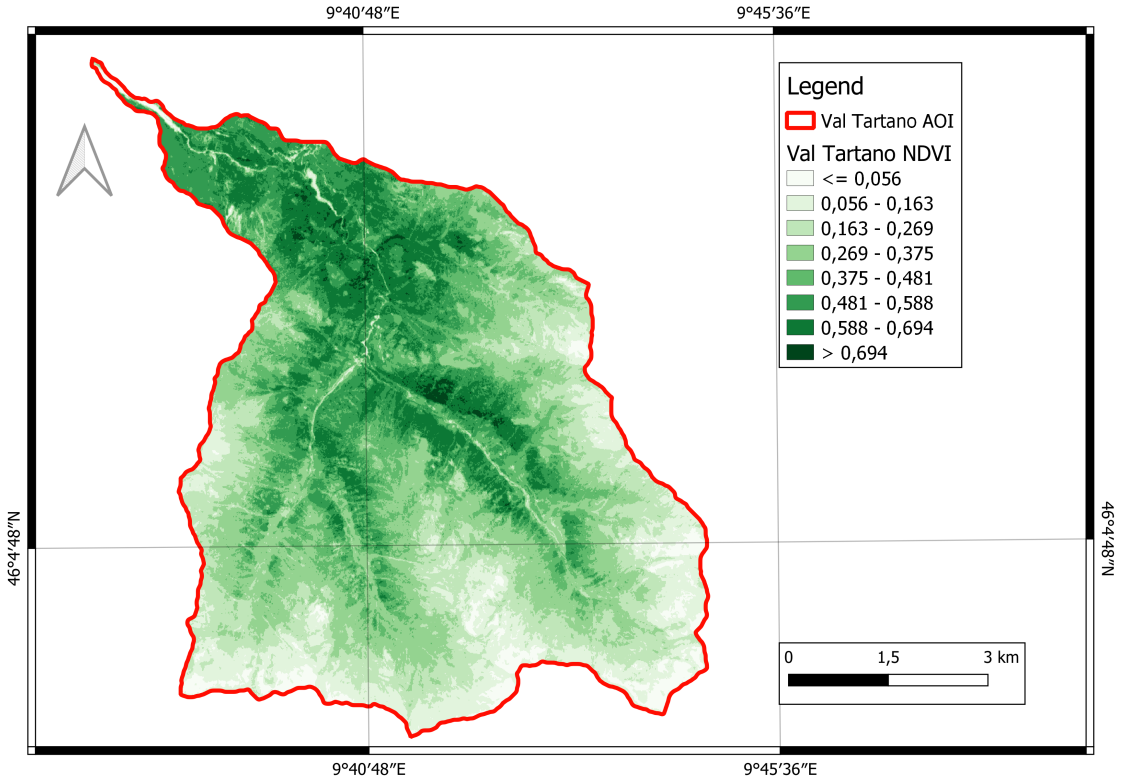
(e) Val Tartano distance from rivers



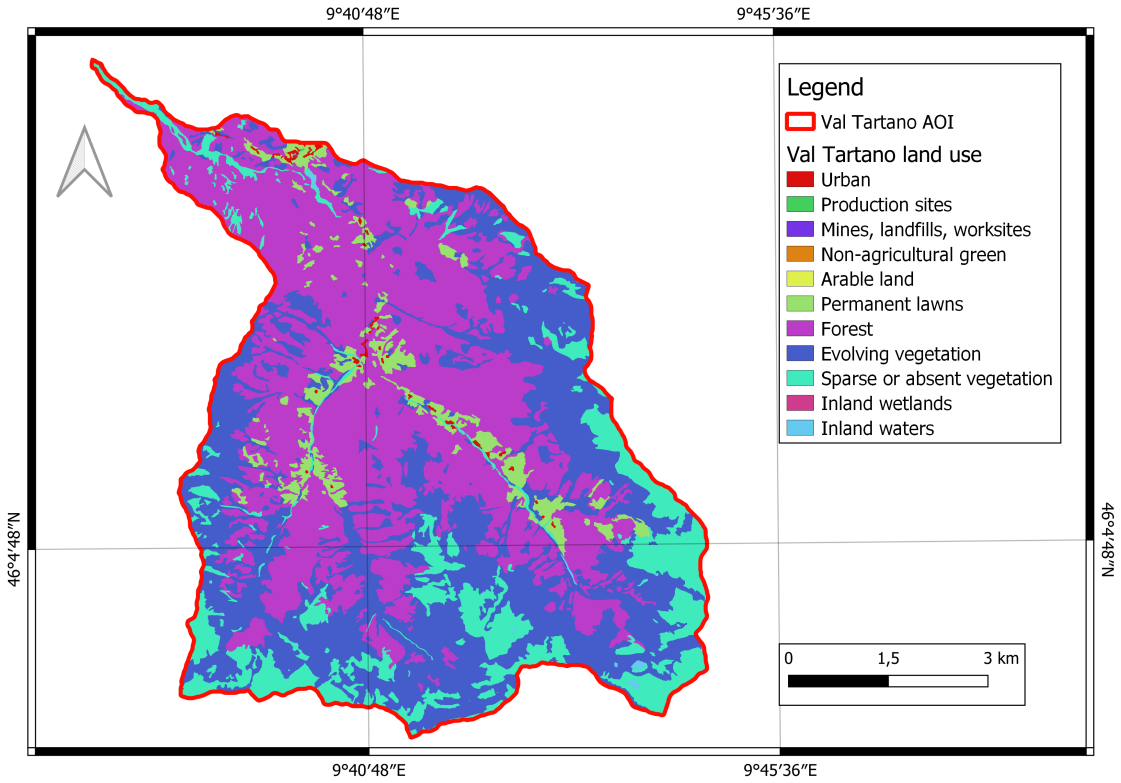
(f) Val Tartano distance from faults



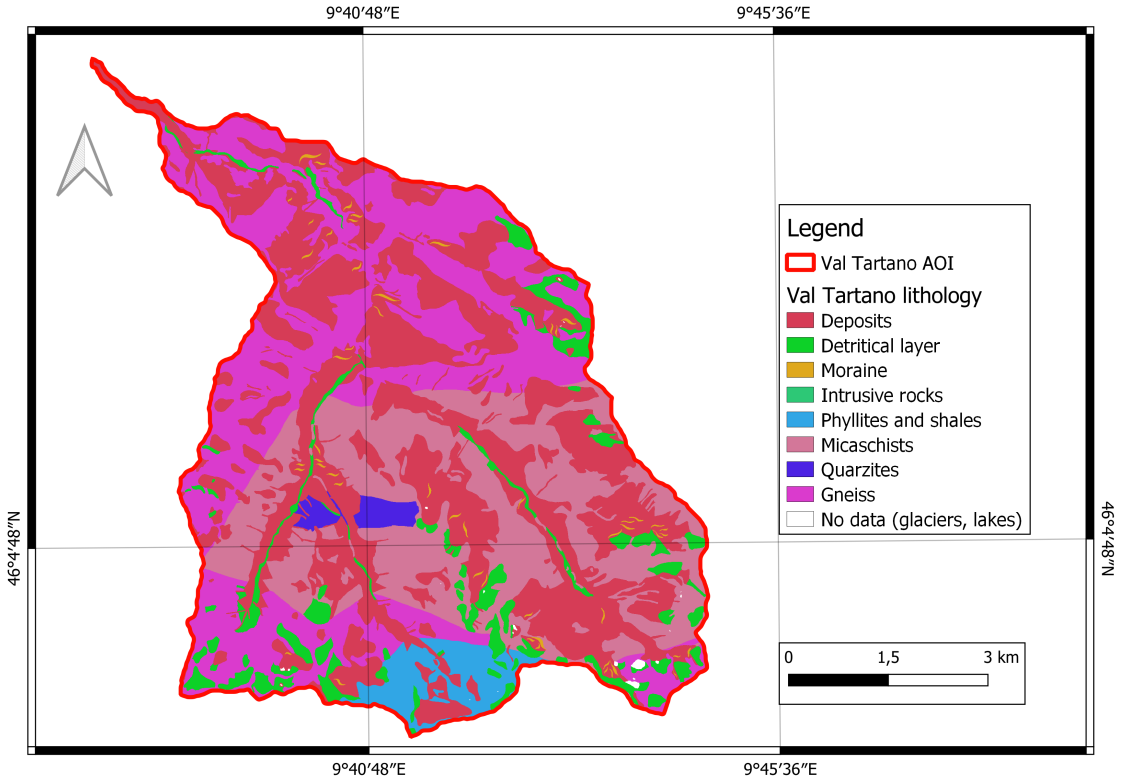
(g) Val Tartano Topographic Wetness Index



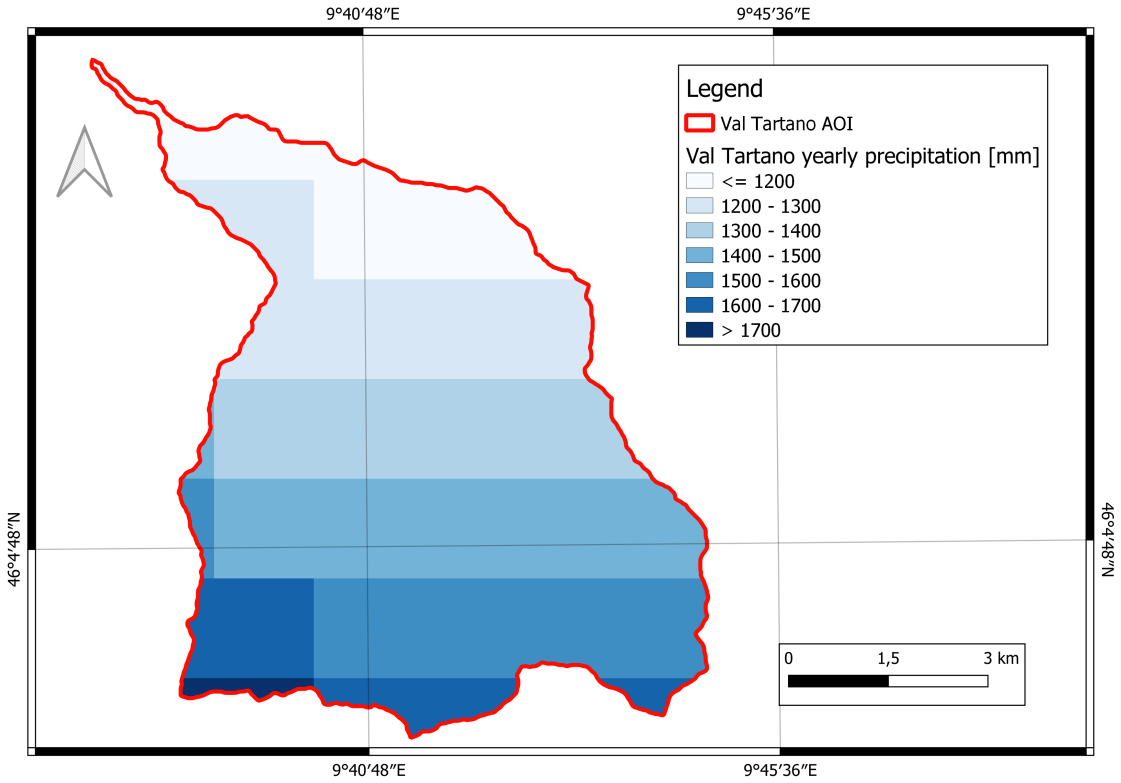
(h) Val Tartano NDVI



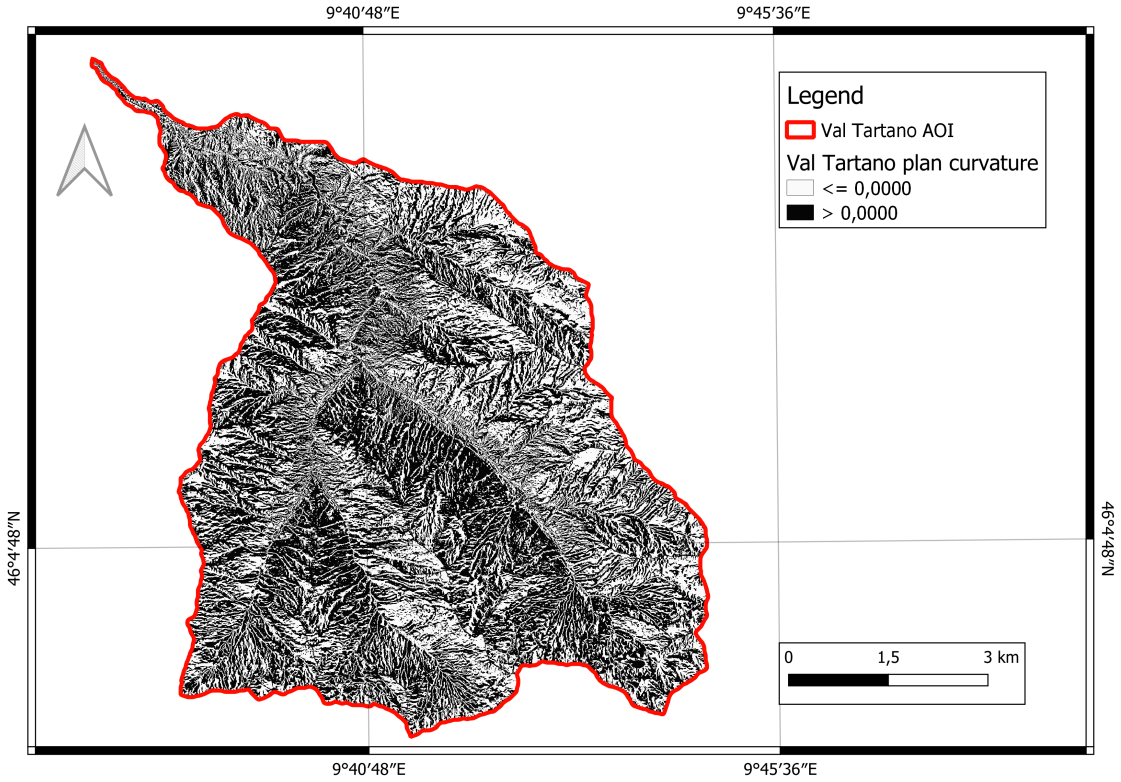
(i) Val Tartano land use



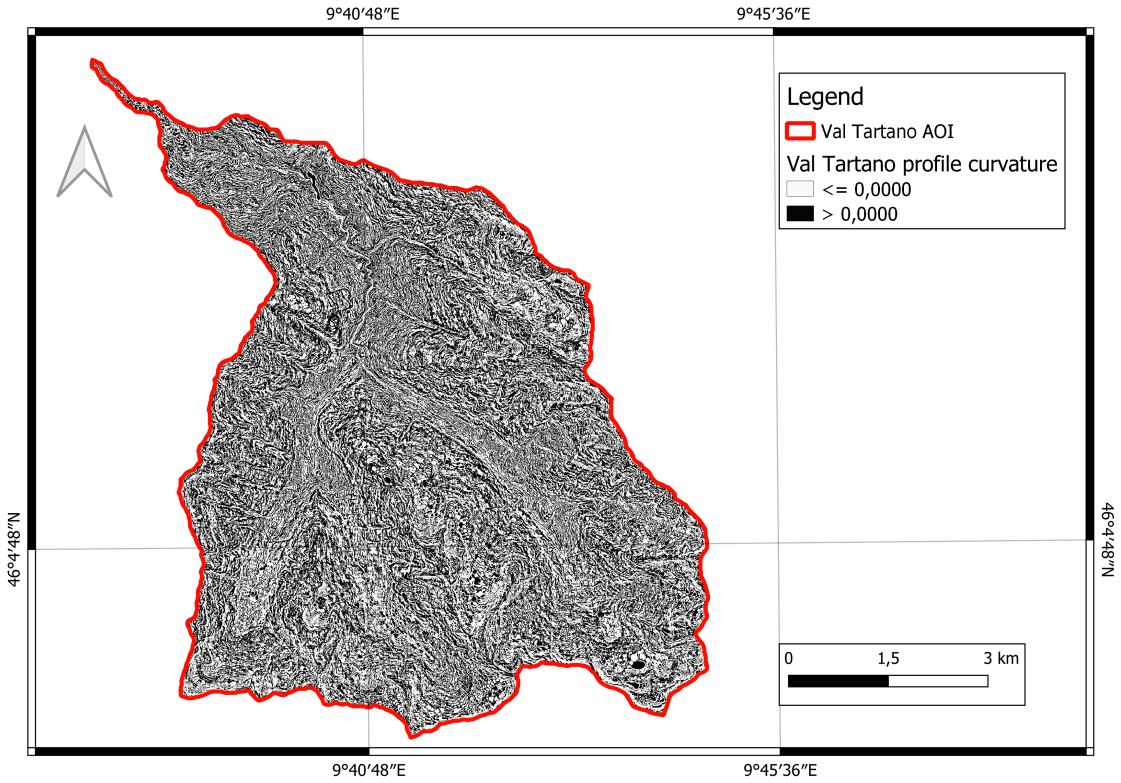
(j) Val Tartano lithology



(k) Val Tartano yearly precipitation



(l) Val Tartano plan curvature



(m) Val Tartano profile curvature

Figure A.2: Val Tartano terrain variables

Appendix B

Code

B.1 Landslide Susceptibility Mapping

B.1.1 NDVI layer computation with Google Earth Engine

```
var ruinon = ee.FeatureCollection('users/lorenzoamici/ruinon_4326');
ruinon = ruinon.geometry();
Map.centerObject(ruinon);
Map.addLayer(ruinon, {color: 'green'}, 'ruinon');

var AOI = AOI_rectangle2;

var AOI_rectangle = AOI.geometries(); // Create a new polygon to clip the image to
AOI_rectangle = AOI_rectangle.get(1);
function maskS2clouds(image) {
  var qa = image.select('QA60');

  // Bits 10 and 11 are clouds and cirrus, respectively.
  var cloudBitMask = 1 << 10;
  var cirrusBitMask = 1 << 11;

  // Both flags should be set to zero, indicating clear conditions.
  var mask = qa.bitwiseAnd(cloudBitMask).eq(0)
      .and(qa.bitwiseAnd(cirrusBitMask).eq(0));

  return image.updateMask(mask).divide(10000);
}

function addNDVI(image) {
  var ndvi = image.normalizedDifference(['B8', 'B4']).rename('NDVI');
  return image.addBands(ndvi);
}
```

```
28
29 var months = [
30     ['2020-01-01', '2020-01-31'],
31     ['2020-02-01', '2020-02-29'],
32     ['2020-03-01', '2020-03-31'],
33     ['2020-04-01', '2020-04-30'],
34     ['2020-05-01', '2020-05-31'],
35     ['2020-06-01', '2020-06-30'],
36     ['2020-07-01', '2020-07-31'],
37     ['2020-08-01', '2020-08-31'],
38     ['2020-09-01', '2020-09-30'],
39     ['2020-10-01', '2020-10-31'],
40     ['2020-11-01', '2020-11-30'],
41     ['2020-12-01', '2020-12-31'],
42 ];
43
44 var visualization = {
45   min: 0.0,
46   max: 0.3,
47   bands: ['B4', 'B3', 'B2'],
48 };
49
50 for(var i = 0; i<12; i++) {
51   var dataset = ee.ImageCollection('COPERNICUS/S2_SR')
52                     .filterDate(months[i][0], months[i][1])
53                     .filterBounds(ruinon)
54                     .filter(ee.Filter.lt('CLOUDY_PIXEL_PERCENTAGE',20));
55
56   // Compute the Normalized Difference Vegetation Index (NDVI).
57   var nir = dataset.mean().select('B8');
58   var red = dataset.mean().select('B4');
59   var ndvi = nir.subtract(red).divide(nir.add(red)).clip(AOI).rename('NDVI');
60
61   var ndviParams = {min: -1, max: 1, palette: ['blue', 'white', 'green']};
62
63   if(i===0) {
64     var NDVIs = ee.ImageCollection(ndvi);
65   } else {
66     var temp = ee.ImageCollection(ndvi);
67     NDVIs = NDVIs.merge(temp);
68   }
69 }
70
71 Map.setCenter(10.47, 46.44, 10);
72
73 var NDVImean = ee.Image(NDVIs.mean());
74
75 Map.addLayer(NDVImean, ndviParams, 'NDVI year mean');
76
77 Export.image.toDrive({
78   image: NDVImean,
79   description: 'imageToDriveExample',
```

```
  scale: 5,
  region: AOI_rectangle
});
```

Listing B.1: NDVI computation code in Google Earth Engine

B.1.2 Precipitation layer computation with Python

```
import os
import numpy as np

sums = np.zeros((174,177))
data_num = np.zeros((174,177))
avgs = np.zeros((174,177))
Initial_directory = "C:\\Users\\lawfr\\Desktop\\Tesi_practice\\PR_2020\\"
Files = []
for file in os.listdir(Initial_directory):
    f = open(Initial_directory + file, 'r')
    lines = f.readlines()
    for j in range(0, 174):
        line = lines[j+6].split()
        for i in range(0,177):
            if (float(line[i])>=0):
                sums[j][i] += float(line[i])
                data_num[j][i] +=1
    f.close()

for j in range(0,174):
    for i in range(0,177):
        if (sums[j][i]==0 and data_num[j][i]==0):
            avgs[j][i] = -9999.0
        else:
            avgs[j][i] = sums[j][i] / data_num[j][i]

avgs = np.asarray(avgs)
np.savetxt(Initial_directory + "precipitation.csv", avgs, delimiter="      ")
```

Listing B.2: Yearly precipitation computation code in Python

B.1.3 ModelMap package implementation for Random Forests landslide susceptibility analysis with R

```
setwd('C:/Users/lawfr/Desktop/Tesi_practice/VAL_TARTANO/LSM')

library('ModelMap')

model.type='RF'

```

```
qdata.trainfn="tartano_training_points.csv"
qdata.testfn="tartano_testing_points.csv"
folder=getwd()

MODELfn="Tartano_20kTP"

predList=c("dtm","rivers","faults","dusaf","roads","profile","plan","north","east",
"twi","ndvi")
predFactor=c("rivers","faults","dusaf","roads")

response.name="hazard"
response.type="continuous"

rastLUT=read.csv("modelLUT.csv")

seed=50
unique.rowname="id"

model.explore(qdata.trainfn = qdata.trainfn, folder = folder, predList = predList,
predFactor = predFactor, response.name = "hazard", response.type = response.type,
device.type=c("jpeg"), response.colors = response.colors,
unique.rowname = unique.rowname, OUTPUTfn = "Tartano_20kTP",units="px",
device.width=8000, device.height=4000, MAXCELL=300000,
res=300, cex=1.5, rastLUTfn = rastLUT, na.value = -9999, col.ramp = heat.colors(101),
col.cat = c("wheat1","springgreen2","darkolivegreen4","darkolivegreen2","yellow",
"thistle2","brown2","brown4"))

model.obj=model.build(model.type=model.type, qdata.trainfn=qdata.trainfn, folder=folder,
unique.rowname=unique.rowname, MODELfn=MODELfn, predList=predList,
predFactor=predFactor, response.name=response.name,
response.type=response.type, seed=seed)

save(model.obj, file="Tartano_20kTP.obj")

model.pred=model.diagnostics(model.obj=model.obj, qdata.trainfn=qdata.trainfn,
qdata.testfn=qdata.testfn, folder=folder, MODELfn=MODELfn, unique.rowname=unique.rowname,
prediction.type="OOB", device.type="pdf", device.width=10, device.height=8,cex=1.2)

model.mapmake(model.obj=model.obj ,folder=folder, MODELfn=MODELfn,
rastLUTfn=rastLUT, na.action="na.omit",map.sd=TRUE)
```

Listing B.3: Landslide susceptibility analysis with the ModelMap package

B.2 Landslide displacement monitoring

B.2.1 Preprocessing of Sentinel-2 images with Python and GRASS GIS

```
import grass.script as grass
import os
from datetime import datetime
import time

start = time.time()

# folder containing the sentinel imagery .SAFE
foldin = 'C:\\Users\\lawfr\\Desktop\\Sentinel2'
# roi Shapefile
roi_in = 'C:\\Users\\lawfr\\Desktop\\Tesi_practice\\Ruinon_monitoring\\aoi_big.shp'
res_in = 10  # GRASS Region target resolution

# BAND LIST
band_list = ["B02", "B03", "B04", "B05",
             "B06", "B07", "B08", "B8A", "B11", "B12"]

folder_out = foldin + '\\' + "outDOS"
if not os.path.exists(folder_out):
    os.makedirs(folder_out)

# SET GRASS REGION
grass.run_command("v.in.ogr", input=roi_in, output="roi", overwrite=True)
grass.run_command("g.region", vector="roi", res=res_in, flags="a")

# GET IMAGES DATES
data = os.listdir(foldin)
dates = []
for item in data:
    if item.endswith('.SAFE'):
        date = (item.split("_")[-1]).split(".SAFE")[0]
        dates.append(datetime.strptime(
            date.partition('T')[0], '%Y%m%d').date())
ldates = list(set(dates))

# PROCESSING SCENES BY DATE
name_list = []
band_name_list = []
for d in ldates:
    for x in data:
        if x.endswith('.SAFE') and x.find(d.strftime("%Y%m%d")) != -1:
            name_list.append(x.split("_")[5]+"_"+x.split("_")[2])

            # IMPORT BANDS (flag -c to import vector cloud mask).
            grass.run_command("i.sentinel.import", input=foldin +
                              '\\'+x, flags="cr", overwrite=True)
```

```
47
48              # CLOUD MASKING
49              cloud_mask_name_original = x.split(
50                  '_')[4]+'_'+x.split('_')[1]+'_MSK_CLOUDS'
51              cloud_mask_name = x.split("_")[5]+"_"+x.split("_")[2]
52              +'_MSK_CLOUDS'
53              cloud_check = grass.read_command("g.list", flags="f", type="vect")
54              if cloud_mask_name_original in cloud_check:
55                  grass.run_command("g.rename", vector="%s,%s" %
56                                    (cloud_mask_name_original, cloud_mask_name))
57                  grass.run_command("v.to.rast", input="%s" % cloud_mask_name, output="%s" %
58                                    cloud_mask_name+'_rast', use='val', overwrite=True)
59                  grass.run_command("g.remove", type='vector',
60                                    pattern="%s" % d.strftime("%Y%m%d"), flags='fr')
61                  for u in band_list:
62                      band = x.split("_")[5]+"_"+x.split("_")[2]+'_'+u
63                      grass.mapcalc("$new = if (isnull($cloudmask)== 1, $original,
64                      null())", new="%s" %
65                                    band+'_cf', cloudmask="%s" % cloud_mask_name+'_rast',
66                                    original="%s" % band, overwrite=True)
67              else:
68                  for u in band_list:
69                      band = x.split("_")[5]+"_"+x.split("_")[2]+'_'+u
70                      grass.run_command("g.rename", rast="%s,%s" %
71                                        (band, band+'_cf'))
72
73      # MERGE SAME BANDS OF MULTIPLE TILES AND CORRECT WITH DOS
74      for n in band_list:
75          band_name_list = [s + "_"+n+'_cf' for s in name_list]
76
77          if len(name_list) > 1:
78              # PATCH BANDS
79              grass.run_command("r.patch", input="%s" % ",".join(
80                  band_name_list), output=d.strftime("%Y%m%d")+"_"+n+'_cf', overwrite=True)
81          else:
82              # SINGLE IMAGERY SCENE, RENAME ONLY WITHOUT PATCHING
83              grass.run_command("g.rename", rast="%s,%s" % (
84                  band_name_list[0], d.strftime("%Y%m%d")+"_"+n+'_cf'))
85          # GET MINIMUM VALUE OF PATCHED BANDS
86          vmin = float(grass.parse_command('r.univar', map='%s' %
87                                           d.strftime("%Y%m%d")+"_"+n+'_cf',
88                                           flags='g')['min'])
89          # COMPUTE DOS
90          grass.mapcalc("$new = $original - $minimum", new=d.strftime("%Y%m%d")+"_dos_"+n,
91                        original=d.strftime("%Y%m%d")+"_"+n+'_cf',
92                        minimum=vmin, overwrite=True)
93
94      # CREATE GROUP
95      name = d.strftime("%Y%m%d")+"_dos"
96      myInput = [name + "_"+n for n in band_list]
97      grass.run_command("i.group", group='s2', input=",".join(myInput))
98
```

```
    # EXPORT TIFF WITH COMPRESSION
    grass.run_command("r.out.gdal", input='s2', output=folder_out +
                      "\\" + name+'.tiff', flags='tc', overwrite=True)

    # REMOVE ALL DATA FROM THE GRASS MAPSET TO START PROCESSING ANOTHER SCENE
    grass.run_command("g.remove", type='raster,vector',
                      pattern="%s" % d.strftime("%Y%m%d"), flags='fr')
    grass.run_command("g.remove", type='group', name='s2', flags='f')
    name_list = []
    band_name_list = []

end = time.time()
print(end-start)
```

Listing B.4: Preprocessing steps for Sentinel-2 Level-1C images

B.2.2 Co-registration of Sentinel-2 images with Python AROSICS package

```
import os
from arosics import COREG
from arosics import COREG_LOCAL
from geoarray import GeoArray
import rasterio as rio

os.chdir(r'C:\Users\lawfr\Desktop\Sentinel2\outDOS')

data = (os.listdir('.'))
files = []
for item in data:
    if item.endswith('.tiff'):
        files.append(item)
files = sorted(list(set(files)))
print('First Image: ', files[0],'\n',
      'Last Image: ', files[-1])
print(files)

for i in range(0,len(files)-1):
    first = f'{files[0]}' # fixed master
    # first = f'{files[i]}' # moving master
    # get a sample numpy array with corresponding geoinformation as reference image

    geoArr  = GeoArray(first)
    print("first" + first)

    ref_ndarray = geoArr[:]

    ref_gt = geoArr.geotransform

    ref_prj = geoArr.projection
```

```
32
33      second = f'{files[i+1]}'
34      # get a sample numpy array with corresponding geoinformation as target image
35
36      geoArr  = GeoArray(second)
37      print("second" + second)
38
39      tgt_ndarray = geoArr[:]
40
41      tgt_gt = geoArr.geotransform
42
43      tgt_prj     = geoArr.projection
44
45      # create in-memory instances of GeoArray from the numpy array data,
46      the GDAL geotransform tuple and the WKT projection string
47      geoArr_reference = GeoArray(ref_ndarray, ref_gt, ref_prj)
48
49      geoArr_target    = GeoArray(tgt_ndarray, tgt_gt, tgt_prj)
50
51  # GLOBAL COREG
52      CR = COREG(geoArr_reference, geoArr_target, path_out=fr'.\aligned\{second}',
53                 max_iter=10000, max_shift=10, match_gsd=False, ws = (100,100),
54                 align_grids=True)
55      CR.calculate_spatial_shifts()
56      CR.correct_shifts()
```

Listing B.5: Co-registration of Sentinel-2 Level-1C images

B.2.3 Clipping of Sentinel-2 images to AOI with Python and GRASS GIS

```
1  import grass.script as grass
2  import os
3  from datetime import datetime
4  import time
5
6  start_tot = time.time()
7
8  # folder containing the sentinel imagery preprocessed
9  foldin = "C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\"
10 # roi Shapefile
11 roi_in = 'C:\\Users\\lawfr\\Desktop\\Tesi_practice\\Ruinon_monitoring\\aoi_small.shp'
12 res_in = 10  # GRASS Region target resolution
13
14 # BAND LIST
15 band_dict = {"B02": "1", "B03": "2", "B04": "3", "B05": "4", "B06": "5",
16              "B07": "6", "B08": "7", "B8A": "8", "B11": "9", "B12": "10"}
17
18 folder_out = foldin + "clipped"
19 if not os.path.exists(folder_out):
```

```
    os.makedirs(folder_out)

# SET GRASS REGION
grass.run_command("v.in.ogr", input=roi_in, output="roi", overwrite=True)
grass.run_command("g.region", vect="roi", res=res_in, flags="a")

# GET IMAGES DATES
data = os.listdir(foldin)
dates = []
for item in data:
    if item.endswith('.tiff'):
        date = (item.split(".")[0]).split("_")[0]
        dates.append(datetime.strptime(
            date.partition('T')[0], '%Y%m%d').date())
ldates = list(sorted(dates))
print(ldates)

# PROCESSING SCENES BY DATE
name_list = []
band_name_list = []
for d in ldates:
    start = time.time()
    for x in data:
        if x.endswith('.tiff') and x.find(d.strftime("%Y%m%d")) != -1:
            imagery_name = (x.split(".")[0]).split("_")[0]
            name_list.append(imagery_name)

            grass.run_command("r.in.gdal", input="%s" % foldin+x, output="%s" %
                              d.strftime("%Y%m%d"), flags='ok', overwrite=True)

            # CREATE GROUP
            myInput = [imagery_name + "."+band_dict.get(n) for n in band_dict]
            grass.run_command("i.group", group='%s' % imagery_name,
                              subgroup='%s' % imagery_name, input=",".join(myInput))

            grass.run_command("r.out.gdal", input='%s' % imagery_name, output='%s' %
                              folder_out+'\\'+imagery_name+'_clip.tiff', flags='tc',
                              overwrite=True)

            grass.run_command("g.remove", type='raster',
                              pattern="%s" % d.strftime("%Y%m%d"), flags='fr')
            grass.run_command("g.remove", type='group',
                              name='%s' % imagery_name, flags='f')

    end = time.time()
    print(end-start)

end_tot = time.time()
print(end_tot-start_tot)
```

Listing B.6: Clipping Sentinel-2 Level-1C images to AOI

B.2.4 Histogram matching of couples of Sentinel-2 images with Python

```
import os
from datetime import datetime
import rasterio as rio
from skimage.exposure import match_histograms
import time

# define paths:
fold_in = "C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped\\"

fold_out = fold_in + "hist_match\\"
if not os.path.exists(fold_out):
    os.makedirs(fold_out)

fold_temp = fold_in + "band_temp\\"

# GET IMAGES DATES
data = os.listdir(fold_in)
dates = []
for item in data:
    if item.endswith('.tiff'):
        date = (item.split(".")[0]).split("_")[0]  # S2
        # date = item.split(".")[0] #L8
        dates.append(datetime.strptime(
            date.partition('T')[0], '%Y%m%d').date())
ldates = list(sorted(dates))

# PROCESS BY DATE AND BAND
for t in range(0, len(ldates)):
    start = time.time()
    t_master = 0
    if t <= max(range(0, len(ldates))) - 1:
        d_t0 = ldates[t_master].strftime("%Y%m%d")
        d_t1 = ldates[t+1].strftime("%Y%m%d")

        # open master image - time 0
        with rio.open("%s" % fold_in+d_t0+"_clip.tiff", 'r+') as r:
            master = r.read()
            metadata_master = r.profile
            metadata_master.update(compress='deflate')

        # open slave image - time 1
        with rio.open("%s" % fold_in+d_t1+"_clip.tiff", 'r+') as s:
            slave = s.read()
            metadata_slave = s.profile
            metadata_slave.update(compress='deflate')

        # store the processed band one by one
        fold_temp_t = fold_temp+d_t1+'\\'
        if not os.path.exists(fold_temp_t):
```

```
            os.makedirs(fold_temp_t)

        # match hist of each band and store results in a new multiband raster
        for band in range(0, slave.shape[0]):

            slave_rescale = match_histograms(
                slave[band], master[band], multichannel=False)

            # Read each layer and write it to stack
            with rio.open("%s" % fold_temp_t+d_t1+"_"+str(band)+"_match.tiff", 'w',
            **metadata_slave) as dstBand:
                metadata_dstBand = dstBand.profile
                metadata_dstBand.update(count=1)
                dstBand.write(slave_rescale, 1)

        print("done image "+d_t1)
        end = time.time()
        print(end-start)
```

Listing B.7: Histogram matching couples of Sentinel-2 Level-1C images

B.2.5 Compressing histogram matched image bands into multiband image with Python and GRASS GIS

```
import grass.script as grass
import os

# folder containing the sentinel imagery preprocessed
foldin_global = 'C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped\\
band_temp\\'
# roi Shapefile
roi_in = 'C:\\Users\\lawfr\\Desktop\\Tesi_practice\\Ruinon_monitoring\\aoi_small.shp'
res_in = 10  # GRASS Region target resolution

folder_out = "C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped\\
hist_match\\"

# SET REGION ACCORDING TO SHAPE AND RES
grass.run_command("v.in.ogr", input=roi_in, output="roi", overwrite=True)
grass.run_command("g.region", vect="roi", res=res_in, flags="a")

filenames = os.listdir(foldin_global)

for date in filenames:
    foldin = foldin_global+date+'\\'
    print(foldin)
    # GET DATES
    data = os.listdir(foldin)

    # PROCESSING SCENES BY DATE
```

```
    name_list = []
    band_name_list = []
    for x in data:
        if x.endswith('match.tiff') and x.find(date) != -1:
            imagery_name = (x.split(".")[0]).split("_")[0]
            bnd_int = int((x.split(".")[0]).split("_")[1])+1
            bnd = "B" + f"{bnd_int:02d}"
            name_list.append(str((x.split(".")[0]).split("_")[0]+'_'+bnd))
            # IMPORT BANDS
            grass.run_command("r.in.gdal", input="%s" % foldin+x, output="%s" % str((
                x.split(".")[0]).split("_")[0]+'_'+bnd), band=1, flags='ok',
                overwrite=True)

    # COMPRESS AND CREATE GROUP
    myInput = name_list
    grass.run_command("i.group", group='%s' % imagery_name,
                      subgroup='%s' % imagery_name, input=",".join(myInput))

    # EXPORT TIFF WITH COMPRESSION
    grass.run_command("r.out.gdal", input='%s' % imagery_name, output=folder_out+
    imagery_name+'_match.tiff', format='GTiff', type="Float64",
    nodata=-1, flags='tcf', overwrite=True)

    # REMOVE ALL DATA FROM THE GRASS MAPSET TO START PROCESSING ANOTHER SCENE
    grass.run_command("g.remove", type='raster,vector',
                      pattern="%s" % date, flags='fr')
    grass.run_command("g.remove", type='group', name='%s' %
                      imagery_name, flags='f')
    name_list = []
    band_name_list = []
```

Listing B.8: Compressing multiple bands output of the histogram matching code into a multiband image

B.2.6 Local cross-correlation of preprocessed images with Python

```
import time
from scipy.spatial import distance
from skimage.feature import match_template
from skimage.registration import phase_cross_correlation
import rasterio as rio
import fiona
import rasterio.mask
import numpy as np
from datetime import datetime
import os
import math

def angle_between(p1, p2):
    deg = math.atan2(p2[1]-p1[1], p2[0]-p1[0])/math.pi*180
```

```
    if deg < 0:
        deg_compass = 360 + deg
    else:
        deg_compass = deg
    return deg_compass

# define template size
row_offset_up = 3
row_offset_down = 4
col_offset_left = 3
col_offset_right = 4


# define step size for moving template
row_step = 1
col_step = 1

# define paths:
fold_in_master = 'C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped\\
hist_match\\'
fold_in_slave = 'C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped\\
hist_match\\'
fold_out = 'C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped
\\hist_match\\final_output_' + \
    str(row_offset_down+row_offset_up) + 'x' + \
    str(col_offset_left+col_offset_right) + '\\'
if not os.path.exists(fold_out):
    os.makedirs(fold_out)

fold_mask = fold_out+"\\masked\\"
if not os.path.exists(fold_mask):
    os.makedirs(fold_mask)

# GET DATES
data = os.listdir(fold_in_master)
dates = []
for item in data:
    if item.endswith('.tiff'):
        date = (item.split(".")[0]).split("_")[0]
        print(date)
        dates.append(datetime.strptime(
            date.partition('T')[0], '%Y%m%d').date())
ldates = list(sorted(dates))

t_master = ldates[0]

for t in range(0, len(ldates)):
    start = time.time()
    if t <= max(range(0, len(ldates))) - 1:
        d_t0 = ldates[t].strftime("%Y%m%d")  # moving master
        # d_t0 = t_master.strftime("%Y%m%d") # fixed master
        d_t1 = ldates[t+1].strftime("%Y%m%d")
```

```
67
68          # open master image - time 0
69          with rio.open("%s" % fold_in_master+d_t0+"_match.tiff", 'r+') as r:
70              master = r.read()
71              metadata_master = r.profile
72              metadata_master.update(dtype=rio.float64, count=1)
73
74          # open slave image - time 1
75          with rio.open("%s" % fold_in_slave+d_t1+"_match.tiff", 'r+') as r:
76              slave = r.read()
77              metadata_slave = r.profile
78              metadata_slave.update(dtype=rio.float64, count=1)
79
80          # define template centres iteratively
81          row_middles_list = list(
82              np.arange(row_offset_up*2, master[0].shape[0]-row_offset_down*2, row_step))
83          col_middles_list = list(
84              np.arange(col_offset_left*2, master[0].shape[1]-col_offset_right*2, col_step))
85
86          # initialize 2darray that contain the new raster bands with matching results
87          distance_array = np.empty((master[0].shape[0], master[0].shape[1]))
88          distance_array[:] = np.nan
89          direction_array = np.empty((master[0].shape[0], master[0].shape[1]))
90          direction_array[:] = np.nan
91          rms_error_array = np.empty((master[0].shape[0], master[0].shape[1]))
92          rms_error_array[:] = np.nan
93          rho_array = np.empty((master[0].shape[0], master[0].shape[1]))
94          rho_array[:] = np.nan
95
96          for row_middle in row_middles_list:
97              for col_middle in col_middles_list:
98
99                  if master[0][row_middle, col_middle] != 0:
100                     # template on the master image
101                     tmpl_row_up = row_middle - row_offset_up
102                     tmpl_row_down = row_middle + row_offset_down
103                     tmpl_col_left = col_middle - col_offset_left
104                     tmpl_col_right = col_middle + col_offset_right
105
106                     template = master[0][tmpl_row_up: tmpl_row_down,
107                                          tmpl_col_left: tmpl_col_right]
108
109                     # template on the slave image
110                     offset_template = slave[0][tmpl_row_up: tmpl_row_down,
111                                                tmpl_col_left: tmpl_col_right]
112
113                     # cross correlation
114                     shift, error, diffphase = phase_cross_correlation(
115                         template, offset_template, upsample_factor=1, space='real')
116
117                     if shift[0] != 0.0 or shift[1] != 0.0:
118
```

```
                        mcorr_offset_template = slave[0][tmpl_row_up+int(shift[0]):
                        tmpl_row_down+int(shift[0]), tmpl_col_left+int(shift[1]):
                        tmpl_col_right+int(shift[1])]

                        # compute distance and direction of the template translation
                        centre = np.array((row_middle, col_middle))
                        match = np.array(
                            (row_middle + shift[0], col_middle + shift[1]))
                        dist = round(distance.euclidean(centre, match), 3)

                        # store change vectors distances, directions, errors and max
                        # norm xcorr coeff (rho) in 2d array
                        distance_array[row_middle, col_middle] = dist
                        direction = round(angle_between(centre, match), 2)
                        direction_array[row_middle, col_middle] = direction
                        rms_error_array[row_middle, col_middle] = error
                        rho = match_template(mcorr_offset_template, template)
                        rho_array[row_middle, col_middle] = rho[0][0]

            if row_middle in list(np.arange(1000, 11000, 1000)):
                print(str(row_middle))

        with fiona.open("C:\\Users\\lawfr\\Desktop\\migration_definitive\\mask.shp", "r")
        as shapefile:
            shapes = [feature["geometry"] for feature in shapefile]

        with rio.open('%s' % fold_out+d_t0+"_"+d_t1+"_distance.tiff", 'w',
        **metadata_slave) as dst1:
            dst1.write(distance_array, 1)

        with rio.open('%s' % fold_out+d_t0+"_"+d_t1+"_direction.tiff", 'w',
        **metadata_slave) as dst2:
            dst2.write(direction_array, 1)

        with rio.open('%s' % fold_out+d_t0+"_"+d_t1+"_error.tiff", 'w',
        **metadata_slave) as dst3:
            dst3.write(rms_error_array, 1)

        with rio.open('%s' % fold_out+d_t0+"_"+d_t1+"_rho.tiff", 'w',
        **metadata_slave) as dst4:
            dst4.write(rho_array, 1)

        with rasterio.open('%s' % fold_out+d_t0+"_"+d_t1+"_distance.tiff") as src1:
            out_image1, out_transform1 = rasterio.mask.mask(
                src1, shapes, crop=True)
            out_meta1 = src1.meta

        out_meta1.update({"driver": "GTiff",
                          "height": out_image1.shape[1],
                          "width": out_image1.shape[2],
                          "transform": out_transform1})
```

```
        with rasterio.open('%s' % fold_mask+d_t0+"_"+d_t1+"_distance.tiff", "w",
        **out_meta1) as dest1:
            dest1.write(out_image1)

        with rasterio.open('%s' % fold_out+d_t0+"_"+d_t1+"_direction.tiff") as src2:
            out_image2, out_transform2 = rasterio.mask.mask(
                src2, shapes, crop=True)
            out_meta2 = src2.meta

        out_meta2.update({"driver": "GTiff",
                          "height": out_image2.shape[1],
                          "width": out_image2.shape[2],
                          "transform": out_transform2})

        with rasterio.open('%s' % fold_mask+d_t0+"_"+d_t1+"_direction.tiff", "w",
        **out_meta2) as dest2:
            dest2.write(out_image2)

        with rasterio.open('%s' % fold_out+d_t0+"_"+d_t1+"_error.tiff") as src3:
            out_image3, out_transform3 = rasterio.mask.mask(
                src3, shapes, crop=True)
            out_meta3 = src3.meta

        out_meta3.update({"driver": "GTiff",
                          "height": out_image3.shape[1],
                          "width": out_image3.shape[2],
                          "transform": out_transform3})

        with rasterio.open('%s' % fold_mask+d_t0+"_"+d_t1+"_error.tiff", "w",
        **out_meta3) as dest3:
            dest3.write(out_image3)

        with rasterio.open('%s' % fold_out+d_t0+"_"+d_t1+"_distance.tiff") as src4:
            out_image4, out_transform4 = rasterio.mask.mask(
                src4, shapes, crop=True)
            out_meta4 = src4.meta

        out_meta4.update({"driver": "GTiff",
                          "height": out_image4.shape[1],
                          "width": out_image4.shape[2],
                          "transform": out_transform4})

        with rasterio.open('%s' % fold_mask+d_t0+"_"+d_t1+"_rho.tiff", "w",
        **out_meta4) as dest4:
            dest4.write(out_image4)

        print('done with '+d_t0+'-'+d_t1)
        end = time.time()
        print(end-start)
```

Listing B.9: Local cross-correlation of preprocessed images to produce displacement maps

B.2.7 Creation of windrose diagrams with Python

```
import os
import numpy as np
import rasterio as rio
import pandas as pd
import matplotlib.pyplot as plt
from windrose import WindroseAxes

fold_in_S2 = 'C:\\Users\\lawfr\\Desktop\\Sentinel2\\outDOS\\aligned\\clipped
\\hist_match\\final_output_7x7_fixed_master\\masked\\'

# set acceptable error
error_threshold = 0.5
error_threshold_text = '05'

period = 'y'
master = 'fixed'

exp_name = '| Template = 7x7 pixels | Moving master '

data_S2 = os.listdir(fold_in_S2)
dates = []
for maps in data_S2:
    date = maps.split('.')[0].split('_')[0]+'_' + \
        maps.split('.')[0].split('_')[1]
    dates.append(date)
ldates_S2 = list(set(dates))

times = []
for t in ldates_S2:
    timeA = pd.to_datetime((t.split('_')[0]))
    timeB = pd.to_datetime((t.split('_')[1]))
    times.append(timeA)
    times.append(timeB)
    times_S2 = list(sorted(times))
start_time_S2 = times_S2[0].strftime("%Y%m%d")
end_time_S2 = times_S2[-1].strftime("%Y%m%d")

# compute variables
d_dis_S2 = []
d_dir_S2 = []
image_count_S2 = 0

for d in ldates_S2:
    # get mask array
    with rio.open("%s" % fold_in_S2+d+'_error.tiff', 'r+') as r:
        mask_image = r.read()
        mask_array_inv = [mask_image[0] > error_threshold]
        mask_array = np.invert(mask_array_inv)
    # read distances and directions
    with rio.open("%s" % fold_in_S2+d+'_distance.tiff', 'r+') as p:
```

```
        dist_image = p.read()
        # mask distances
    dist_image[0][mask_array[0] == False] = np.nan
    dist_image[0][dist_image[0] == -1] = np.nan
    # read distances and directions
    with rio.open("%s" % fold_in_S2+d+'_direction.tiff', 'r+') as s:
        dir_image = s.read()
    # mask directions
    dir_image[0][mask_array[0] == False] = np.nan
    dir_image[0][dir_image[0] == -1] = np.nan
    # Create landslide distance and direction variables and plot them
    d_dis_S2.append(dist_image[0].flatten()[
                    np.logical_not(np.isnan(dist_image[0].flatten()))]*10)
    d_dir_S2.append(dir_image[0].flatten()[
                    np.logical_not(np.isnan(dir_image[0].flatten()))])

    image_count_S2 += 1

dist_array_S2 = np.concatenate(d_dis_S2, axis=0)
dir_array_S2 = np.concatenate(d_dir_S2, axis=0)

n_of_pixels_S2 = int(dir_image[0].shape[0]*dir_image[0].shape[1])
n_of_changes_S2 = int(dist_array_S2.shape[0])
textstr = '\n'.join((
    r'$error-level=%.2f$' % (error_threshold, ),
    r'$T_{start}=%s$' % (start_time_S2, ),
    r'$T_{end}=%s$' % (end_time_S2, )))

# Plot distance and direction variables
ax = WindroseAxes.from_ax()
ax.set_xticklabels(['E', 'N-E', 'N', 'N-W', 'W', 'S-W', 'S', 'S-E'])
ax.bar(dir_array_S2, dist_array_S2, normed=True,
       opening=0.8, edgecolor='white')  # nsector=16
ax.set_legend(title="Displacement [m]", bbox_to_anchor=(
    1, 0), loc="lower right", bbox_transform=plt.gcf().transFigure)
props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)
ax.text(0.0, 0.95, textstr, transform=ax.transAxes, fontsize=10,
        verticalalignment='bottom', bbox=props)
plt.savefig(fold_out+period+'_'+master+'_windrose.png')
```

Listing B.10: Creation of windrose diagrams in order to summarize results obtained from the monitoring analysis