UNIVERSITY OF CATANIA
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
PhD IN COMPUTER SCIENCE (INTERNATIONAL) - XXXV CYCLE

ROBERTO SPINA

COMPUTER TECHNIQUES APPLIED TO ENVIRONMENTAL MONITORING

PhD THESIS

TUTOR: CHIAR.MO PROF. EMILIANO TRAMONTANA
COORDINATOR: CHIAR.MO PROF. SEBASTIANO BATTIATO

Giacché il tessuto dell'universo è perfettissimo e opera di un Creatore sapientissimo, nulla in assoluto nell'universo accade che non riveli un processo di massimizzazione o minimizzazione.

Leonhard Euler (1707 -1783)

Opera I 24, p.231

Abstract

Environmental monitoring is a territory control technique essential in preventing destructive natural phenomena. The fragile conditions of the topographical reliefs, the extreme climatic conditions and the increasing pollution of the hydrosphere and lithosphere accelerate terrestrial areas' degradation. These phenomena are greatly affected by the anthropic action that amplifies their characteristics, contributing to the precariousness of the living conditions of terrestrial biomes.

The upheavals of the terrestrial lithosphere brought about by extreme events due to endogenous (volcanic eruptions and earthquakes) and exogenous (floods, tsunamis, landslides) phenomena, fit examples. In this context, adopting strategies capable of mitigating the effects produced on the environment and predicting extreme natural events in advance is of great importance. In both cases, these techniques have a twofold objective: to reduce the loss of human life and preserve natural resources.

The works included in this thesis show different approaches focused on specific cases. We aim for alternative proposals that make it possible to reduce the existing gap in the prevention and monitoring of natural phenomena. The prevention of seismic events based on precursory phenomena currently has a high degree of uncertainty. The proposed approach is based on the threshold values of each precursor related to paroxysmal seismic or volcanic events and could significantly increase their reliability. Moreover, in satellite imagery-based environmental monitoring, some steps are usually performed through the GUI of Geographic Information System (GIS) applications. More advanced technologies should be included to automate as much as possible the analysis. This work proposed a fully automated strategy to monitor urban sprawl and heat fields in the Etna volcano and to evaluate their evolution in space and time. Extensive experiments on images depicting the areas of several cities have shown the reliability of the proposed approach and tools.

Contents

Chapter 1

Introduction

Environmental monitoring can have different purposes, however the main objective is always to protect human and natural resources from events that could compromise their integrity. There is currently no model capable of predicting earthquakes effectively in the seismic field [1, 2, 3]. Over the years, the forecasts have not been satisfactory, and the results have shown large deviations from those expected, deduced from the modifications undergone in the precursor parameters (seismic swarms, ground deformations, radon gas emissions, volcanic tremor, etc.). Similar behaviour was also had in the volcanic field [4, 5]. The study suggested a possible interpretation to explain the results that do not comply with the seismic and volcanic forecasts. An integrated analysis of all available precursors based on Boolean constructs could provide more reliable results than those based on single precursors.

Monitoring can also be performed passively, without using precursor phenomena and sensors that measure their variations in space and time. The proposed method involves using satellite images and analysis techniques through image processing. In this way, it is possible to quantify the size of green areas within urban centres, providing crucial information on the evolution of urban sprawl. This phenomenon, due to an irrational expansion of urban areas at the expense of green spaces, is an indicator of city healthiness and quality of life.

Monitoring of some environmental phenomena can be done through an au-

tomated analysis of variations in space and time. The same image processing techniques used previously can be applied to image swarms to evaluate the variations of specific parameters (green and urban surfaces, soil temperature) occurring in a specific time interval. This technique allows us to interpret, qualitatively and quantitatively, the evolution of some natural phenomena both in the medium-long term (urban sprawl) and in the short term (variation of heat cells in volcanic areas, etc.). Each of these adopted techniques requires performances that allow significant amounts of information to be processed in a reasonable time.

Strategies based on refactoring the parts of code with greater computational complexity, refactoring to microservices with external parallelization of specific containers and hybridization of imperative programming with functional matrix constructs are some solutions adopted for this purpose.

The following sections introduce the salient features of the topics covered by the work carried out in the three-year doctorate and their structuring.

1.1 A novel approach for the forecast of seismic and volcanic events

The forecast of imminent seismic or volcanic events can drastically reduce the loss of human life by alerting the inhabitants living in the surrounding areas. Seismic Early Warning (SEW) based on the acquisition-transmission of the main seismic precursors (seismic swarms, soil temperature and deformation, radon gas emissions, etc.) [6, 7], implemented on efficient software infrastructure can help reduce the time required to forecast high-intensity seismic and volcanic events.

By exploiting the considerable amount of data acquired by the sensors, it is also possible to simulate some typical scenarios that are a prelude to an earthquake or a volcanic eruption, for example, by predicting the path of a lava flow [8] or the energy released by a possible event. In this context, parallel computing on GPU [8, 9, 10] allows the processing of large amounts of data in a relatively short time,

compared to the traditional CPU-based approach, with undoubted advantages for decision-making timing in the event of possible earthquakes or catastrophic eruptions. One of the purposes of this work is to help reduce the gap in seismic and volcanic forecast by creating innovative algorithms based on Boolean expressions and using GIS techniques supported by the analysis of satellite images. The first approach is based on the data provided by some geophysical parameters called seismic and volcanic precursors.

When a disastrous earthquake is about to occur in a specific territory, there are a series of anomalies that alter the pre-existing natural balances. Seismic swarms, ground deformation, bright flashes, emissions of various gas types (radon, CO2,etc.), changes in the composition and flow rate groundwater are just some physical-chemical perturbations induced by the growing stress condition borne by the crustal masses. Dilatancy theory and asperity or barrier model allow us to interpret the dynamic mechanisms to which the seismic precursors are due: the development of a network of cracks and the sliding of areas with less mechanical resistance are in agreement with seismic, mechanical and geochemical anomalies that occur before to high magnitude earthquakes. In areas with high seismic risk, constant monitoring of geophysical parameters is frequent, carried out using different types of sensors.

The MAS (Multi-Agent System) model is one of the most suitable choices for efficiently implementing a seismic alert system, based on the interpretation of experimental data obtained from the sensor network. Using this type of approach, a Seismic Early Warning (SEW) has been created that according to the data acquired by the sensors and through the activities carried out by agent clusters, define the risk of seismic events having magnitude at least six. The SEW system aims to interpret, in real-time, the variations of an adequate number of seismic precursors for specific threshold values, calculated statistically. The integrated and complementary analysis of them, using several specific Boolean expressions, assesses the contribution provided by each parameter for computing the level of risk, divided into soft, medium and hard. The model has been tested with data

gathered in New Zealand, a nation with a high seismic and volcanic risk which offers free access to some seismic precursors.

1.2 Urban sprawl control by image processing

The irregular and uncontrolled expansion of urbanized areas, defined as urban sprawl [11, 12, 13, 14, 15], is one of the main environmental challenges that have significant impacts on the quality of life and economic performance of cities. As a result, there is an abandonment of agricultural activities and urban sprawl into agricultural land, sometimes of high quality [16, 17]. In contrast to this common trend, several cities around the world have promoted different policies to manage the growth of urban areas and protect natural and agricultural areas [18, 19]. An example is the USA, where green regulation has become a fundamental factor for the management of urban spaces, involving both public and private ownership [20, 21].

In this context the delimitation of green areas from urban ones is a fundamental operation for a correct estimate of green spaces within cities. Specific GIS applications assist users in the tracing of boundaries and the calculation of lengths and areas. However, when using such techniques, the main limitation concerns the analysis of irregularly shaped areas. For them, tracing outlines that overlap the original ones is performed manually, hence difficult, error prone and time consuming.

We propose a novel approach that is automatically performed by means of a Python-based application to recognize boundaries and compute the main geometric parameters of urban and green areas. Such operations are based on new algorithms that give an innovative character to the entire application tested in Kamakura and Acireale territories. The first is a Japanese city that embodies the "City Country Fingers" design pattern characterized by extensive intersections between the vegetation from the peripheral areas and the urban centre. Their complexity and irregularity are a proper testbed to evaluate the behavior of the application

in the presence of irregular areas.

Unlike the previous case, in the Acireale city, the country fingers have a mainly two-dimensional development with the corresponding city fingers surrounded by agricultural land mostly used for the cultivation of citrus fruits (oranges, tangerines and lemons) and low-stemmed plants (vine). The progradation of this type of vegetation cover does not present any structural recursion with country fingers that develop linearly toward the urban center.

1.3 Space-time analysis of natural phenomena

Natural phenomena, both extreme and low-medium entities, show variable characteristics in space and time. In some cases, it is essential to study the variations that occurred in a given time interval. The rate at which specific parameters vary can provide helpful information on the evolution of the phenomenon in progress and the degree of risk to which the territory in question is subject.

In the seismic and volcanic fields, the space-time increase of some geophysical parameters is evaluated through in situ measurements. In other cases where the evolution of natural phenomena is influenced by anthropogenic activity, it is necessary to use other methods to get an idea of how they vary in space and time. An example is represented by the phenomenon of urban sprawl, which we talked about in the previous section and directly impacts cities' livability. Another natural phenomenon which offers significant points of interest is the analysis of thermal anomalies in volcanic areas, which has been treated as an application example of the adopted method.

A technique that is well suited to both purely natural phenomena and those influenced by human activity is represented by monitoring using satellite images. Analyzing a set of them relating to a specific time interval allows us to trace the changes that occurred in that time frame and a particular sector of the earth's crust. An analysis of this type is of particular importance in the GIS field as it allows, in an automated way and in very few instants of time, to define the

conditions of the territory under different aspects (seismic, volcanic, hydrographic, environmental, etc.).

Our proposal, therefore, concerns a powerful multitasking approach capable of monitoring natural phenomena and those related to the sustainable development of the territory.

1.4 Code refactoring to improve the performance of GIS applications

The current monolithic applications have limitations in performing single or multiple images at increasing resolution. The temporal study of some phenomena may require processing applied to a swarm of them. We mainly refer to the geospatial analysis applied to urban agglomerations to evaluate the decrease in green spaces or similar phenomena that require comparing satellite imagery relating to different periods. The study of the desertification process is another example that necessitates the simultaneous processing of image swarms. The algorithms performed during processing require adequate computational resources, especially those that have to scan, pixel by pixel, the entire image or calculate the maximum and minimum values of x for the same y and vice versa [22]. Therefore, improving performance to large input loads or single images of medium-high resolution is necessary.

In this case, many architectural and implementation choices must be made innovatively and strategically for optimal problem resolution. The external parallelization of containers and the exchange of data through an asynchronous messaging system requires their "ad hoc" use. The transformation of the initial monolithic system into a microservices one makes possible some types of analysis in different fields (urban planning and geophysics, to name a few) that were previously not possible for the time required. Performance analysis and evaluation of flexibility and portability can give valuable indications of the best compatibility between the

two systems.

1.5 Thesis structure

Chapter 2 discusses a new algorithm for implementing a seismic-volcanic warning system. It comprises different sections, starting from the theoretical background of Seismic Early Warning, framework and implementation details and hardware and software architecture. The experiment was carried out on the territory of New Zealand, which due to its particular geodynamic position, was chosen as a case study. The availability of the data, represented by earthquakes and ground deformations, made it possible to test the method's reliability.

Chapter 3 concerns environmental monitoring through image processing techniques applied to satellite imagery. This way, automated analysis is carried out to reach significant results described in the different sections. Labelling boundaries and surfaces and parameterizing green and urban areas are the two main phases that allow qualitative and quantitative monitoring. The data obtained from the geospatial analysis allowed us to perform queries in JSON format to extract helpful information from the green and urban areas.

In Chapter 4, the previous techniques used to perform the parameterization of green and urban areas have been extended to a swarm of images. The main purpose is to carry out a space-time analysis that allows us to obtain, in various steps, a map of the gradients of both areas in different urban centres. The method's reliability has been tested on two Japanese cities, Kamakura and Ōhata, which represent an excellent testbed due to the articulation of the urban green.

Chapter 5 describes the techniques applied to obtain the refactoring of the original application. The main strategies adopted are: rewriting portions of code from imperative to functional syntax, implementing the original monolithic application in microservices, and parallelizing containerized algorithms.

Finally, Chapter 6 provides the conclusion of the thesis, and the following sections concern the lists of figures and tables, and the bibliographic references

used within the document.

1.6 Published papers

Part of the work presented in this thesis is based on the following co-authored papers:

- R. Spina, A. Fornaia, E. Tramontana: *VSEW: an early warning system for volcanic and seismic events.* **Proceedings** of IEEE International Conference on Smart Computing, SMARTCOMP, 2020.

- R. Spina, A. Fornaia, E. Tramontana: *An Early Warning System for Seismic Events based on the Multi-Agent Model.* **Proceedings** of the WOA Workshop "From Objects to Agents", 2020.

- R. Spina, E. Tramontana: *An Image-Processing Approach for Computing the Size of Green Areas in Cities.* **Proceedings** of the 9th ACM International Conference on Computer and Communications Management (ICCCM), 2021.

- R. Spina, E. Tramontana: *An automated classification system for urban areas matching the 'city country fingers' pattern: the cases of Kamakura (Japan) and Acireale (Italy) cities.* **Journal of Urban Ecology, Oxford University Press**, Volume 7, Issue 1, 2021.

- A. Fornaia, S. La Torre, G. Pappalardo, R. Spina, E. Tramontana: *A novel approach for geospatial analysis in raster mode based on microservices and data streaming* **In review**.

- R. Spina, E. Tramontana: *An example of GIS applications to perform a spatio-temporal analysis of a swarm of satellite imagery.* **Advanced draft**.

Chapter 2

The seismic and volcanic early warning systems

2.1 Approach

In areas with high seismic risk, the mainshock is preceded, in time intervals varying from days to years, by the so-called "foreshock", swarms of seismic events that can prelude to the breakage or reactivation of the discontinuity surfaces (faults) present in the earth's crust that generate the main event (mainshock). Often, areas with high seismic risk are embedded in one or more active volcanoes, as in the case of the Pacific "Ring of Fire" in which about 90% of the world's earthquakes occur. High intensity seismic and volcanic events are generally preceded by significant variations of the physical-chemical parameters, which indicate potential instability of the territory. In some "asymptomatic" cases, the seismic or volcanic event is not however preceded by important anomalies of the geophysical precursors. The main seismic precursors are represented by the emission of radon gas, soil deformation, soil temperature and the geomagnetic and gravitational anomalies related respectively to the variations of the magnetic and gravimetric field before an event of a certain intensity. In the volcanic field, the main precursors are represented by volcanic tremor, a seismic signal of weak intensity induced by vibrations due to the rise of magma, soil deformation, thermography, concentration of volcanic

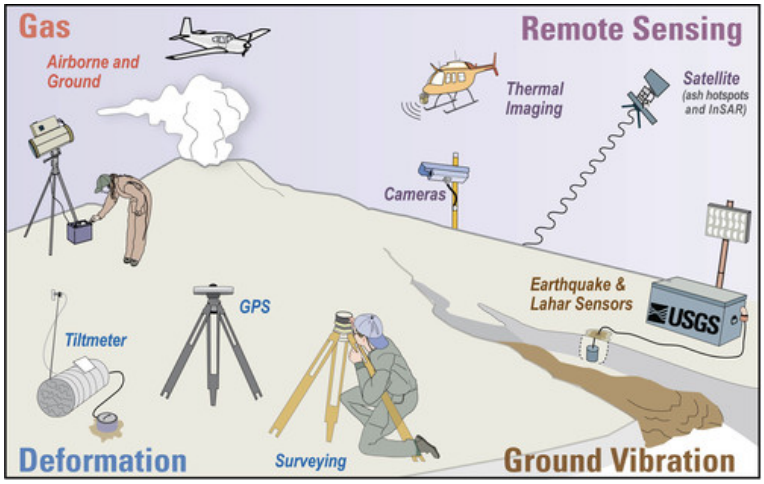


Figure 2.1: The USGS measures the activity level of a volcano with several different types of instruments. This graphic represents types of volcano monitoring in the corners, with associated methods used in italics (Credit: Lisa Faust, USGS).

gases and presence of seismic swarms that can give valuable information on the opening of new volcanic vents (Figure 2.1). In addition to terrestrial detection, satellite imagery also allow us to observe and obtain useful information on the evolution of some precursor phenomena: thermal anomalies, volcanic plumes and soil deformations.

Overall, the precursor phenomena generate an enormous amount of data acquired by the respective sensors, which require an adequate structuring of the software for their management. The MAS (Multi-Agent System) model is one of the most suitable choices for efficiently implementing a seismic alert system, based on the interpretation of experimental data obtained from the sensor network. Software applications based on MAOP (Multi-Agent Oriented Programming) are widely used in various fields and have taken on an increasingly important role thanks to the use of artificial intelligence (AI) techniques [23, 24, 25].

The adoption of centralized methods presents intrinsic difficulties due to the growing complexity of the systems, the dimensions of which continue to increase: in this context, the architectural solutions proposed by the MAS (Multi-Agent System) offer different advantages and a good solution for the modelling of complex distributed systems [25, 26, 27]. One of the fundamental characteristics of the MAS

paradigm is the interaction between agents, independent and autonomous software modules that perform specific propaedeutics activities for the development of one or more system functions. The innumerable properties that distinguish the agents (communication, persistence, reactivity, proactivity, etc.) make them potentially suitable for monitoring natural phenomena, especially those capable of producing catastrophic events. The different activities required by a seismic early warning system, summarized in the acquisition, interpretation and formatting of geophysical data with the addition of real-time user assistance, make the MAS model one of the most appropriate systems for real-time monitoring of precursor parameters (seismic swarms, ground deformation, soil temperature, radon gas emission, etc.) aimed at predicting earthquakes potentially destructive.

2.2 Related Work

Monitoring procedures of geophysical parameters, in seismic and volcanic areas, still require human intervention which often involves significant delays between the occurrence of an eruption and notifications being dispatched. A group of scientists presented the first examples of operational early-warning for volcanic eruptions based on automatic and unsupervised algorithm. The system consists of an infrasound array that identifies and automatically transmits to the authorities, in real time, the imminent occurrence of an explosive eruption [28]. Other alert systems have been created using different techniques and algorithms [29, 30, 31].

Different geophysical parameters have been used for several years to predict earthquakes and eruptions with contrasting results. The use of radon-thoron isotope pair is considered a new opportunity for earthquake forecasting: unusually large ^{220}Rn peaks (the decay product of radium-226) were observed only in February 2011, preceding the 2011 M 9.0 Tohoku-Oki Earthquake [32]. Soil temperature and seismic swarms have also proven, on several occasions, to be reliable precursors of seismic [6], [7] and volcanic events [33], [34]. A promising technology, used for seismic precursors, is represented by the InSAR satellite techniques for the

analysis of soil deformations which allowed the detection, three years before the 2009 earthquake in the Aquila city, of an extensive subsidence of the soil (up to 15 cm) and a post-earthquake lift of about 12 cm [35].[6], [7] and volcanic events [33], [34]. A promising technology, used for seismic precursors, is represented by the InSAR satellite techniques for the analysis of soil deformations which allowed the detection, three years before the 2009 earthquake in the Aquila city, of an extensive subsidence of the soil (up to 15 cm) and a post-earthquake lift of about 12 cm [35].

Many applications that combine artificial intelligence and MAS technologies, in various sectors, have been developed: some examples are represented by the new opportunities created respectively for traffic control at intersections [23], for monitoring and improving the cloud performance and security [24], or for alerting people about crowded destinations [36]. In the last years, a group of scientists presented a Multi-Agent System paradigm and discuss how it can be used to design intelligent and distributed systems [25]. Next, a decentralized approach of MAS has been developed using a distributed simulation kernel to solve partitioning, load balancing and interest management problems in an integrated, transparent and adaptive manner [26].

Different works have been produced on the implementation of multi-agent systems relating to coordination and rescue in the stages following the occurrence of a high-intensity seismic event. A multi-agent system for the evacuation of people in immediate post-emergency situations has been implemented for the city of Iaşi (Romania) [37]. A series of simulators using a MAS architecture, following the damage caused by the 1999 earthquakes in Turkey and Pakistan in 2005, have been developed: damage, victims and other auxiliary simulators [38]. A Disaster Management System (DMS) developed with the multi-agent model has been proposed to adequately manage a multi-risk situation consisting of two or more disasters occur at the same time, such as, for example, the combination of earthquake and tsunami [39].

Other systems for the management of the pre-post seismic phases have been

developed by the authors in various ways: (i) through a Seismic and Volcanic Early Warning System in the Etna area based on specific threshold values for each geophysical precursor [40]; (ii) with an approach based on the coupling of multi-agent systems and intelligent systems (cellular automata) for simulation on rescue in the event of an earthquake disaster [41]; (iii) through simulations of various post-seismic evacuation scenarios for people using a multi-agent system [42]; (iv) integrating GIS with multi-agent seismic disaster simulations to investigate factors significantly affecting rescue efforts, and to clarify countermeasures for saving lives [43].

2.3 Theoretical Background of the Seismic Early Warning

2.3.1 The barrier and asperity models

Forecasting of high-magnitude seismic events has as its foundation some theories that, since the last century, have been proposed by various authors to explain the phenomena that determine earthquakes.

Dilatancy theory [44] foresees that before an earthquake the seismogenic area is subject to an increase in stress with an expansion of the crustal volume due to a substantial cracking of the rocks. Consequently, the rocks undergo a variation of their physical characteristics and from the external regions, the fluids are attracted by this extensive fracturing phenomenon. Both the gases and the liquids circulating within the crustal volume change their paths and upon contact with different rocks and/or fluids change their geochemical composition. The interpretation of the phenomena that prelude and follow an earthquake is the basis of what is proposed by Aki (1979) and Kanamori (1981) called respectively barrier model and asperity model.

In the *barrier model* [45, 46] it is assumed that, before the earthquake, the stress on the fault is uniform. The earthquake is produced by the sliding of the

weakest area, while the most resistant area (barrier) is opposed to dislocation. In this way, there is an increase in barrier stress. Consequently, after the earthquake, the barrier may be affected by seismic (aftershock) or aseismic sliding episodes.

The *asperity model* [45, 46] considers that the sliding that generates the earthquake concerns the most resistant area, i.e. the asperity. Before the mainshock, the stress on the fault is not uniform because aseismic sliding and preliminary shocks (foreshock) have reduced stress in the weakest areas of the fault, concentrating it on asperities. When stress reaches a critical value, the asperity yields giving rise to the earthquake.

The three models agree with the physico-chemical anomalies related to the extensive fracture affecting the seismogenic area (seismic precursors) and with the long sequences of earthquakes preceding (foreshock) and subsequent (aftershock) at the mainshock, providing a valid interpretative key.

2.3.2 Statistical basis

Within each seismic zone, there are one or more seismogenic structures (faults) which, with their displacement, can produce the vibrations that generate the earthquake. Sequences of seismic events that, in some cases, may prelude to a major magnitude earthquake (mainshock) are called seismic swarms. Every single event (foreshock) belonging to the sequence often occurs a short time from the previous one.

Suppose we consider all the seismic swarms that in the past have given rise to mainshocks of medium-high magnitude (above 6) which, about to the characteristics of the territory concerned, can produce serious damage to people and things. We denote with S_1, S_2, S_3 three classes of seismic swarms (SS) which as a final result gave an earthquake of magnitude $M_w \geq 6$ and with (P_1, $\bar{S}_1$), (P_2, $\bar{S}_2$), (P_3, $\bar{S}_3$), the ordered pairs where P corresponds to the number of S elements and $\bar{S}$ to the arithmetic mean for each class S_1, S_2, S_3. The average of the averages for the three classes of seismic swarms that prelude to an earthquake of Magnitude M_w

will be given by:

$$SS_{Th}=\frac{1}{N}\sum_{i=1}^{3} P_i \bar{S}_i \qquad (1)$$

$$N = P_1 + P_2 + P_3. \qquad (2)$$

The SS_{Th} value obtained corresponds to the most probable value for seismic swarms with $\bar{M}_d < 6$ for that specific seismogenic structure, and therefore a threshold value for the mainshock of magnitude $M_w \geq 6$.

The same procedure is performed for geophysical precursors for which a consistent database of measurements is available. Consequently, three further threshold values (RC_{Th}, GD_{Th} and ST_{Th}) will be obtained referring respectively to Radon Concentration (RC), Ground Deformation (GD) and Soil Temperature (ST) for earthquakes with $M_w \geq 6$. The four previously threshold values (T_h) will be associated with the respective standard deviations (σ) expressed by:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{3}(\bar{S}_i - T_h)^2}{3}} \qquad (3)$$

and the achievement of the threshold value (T_h) will occur when:

$$[\bar{M}_w \pm \sigma_w] \cap [SS_{Th} \pm \sigma_{Th}] = \emptyset. \qquad (4)$$

where $\bar{M}_w$ represents the average magnitude of the current seismic swarm and σ_w the standard deviation associated with it. In real conditions in presence of an extensive seismic swarm (SS), the system will calculate the average magnitude value ($\bar{SS}$) of the seismic sequence in progress and the other three seismic precursors ($\bar{RC}$, $\bar{GD}$ and $\bar{ST}$). In the next step, it will compare the mean value of the four precursors with the respective threshold values.

2.3.3 Boolean expressions based on precursor phenomena

The definition of the alarm level is based on the evaluation of a series of Boolean expressions and conditional instructions, arranged in sequence, which allows defining the type of alarm based on the number of precursor parameters that have reached or exceeded the threshold value, going from hard (all variables are true) to soft (only two variables have exceeded the threshold value). Among the four Boolean variables, SS (Seismic Swarm) has a fundamental role in defining any alarm level:

$$
\begin{aligned}
&if(SS \wedge RC \wedge GD \wedge ST) \\
&\qquad \text{return "hard";} \\
&else\, if(SS \wedge ((RC \wedge GD) \vee (RC \wedge ST) \vee (GD \wedge ST))) \\
&\qquad \text{return "medium";} \\
&else\, if(SS \wedge (RC \vee GD \vee ST)) \\
&\qquad \text{return "soft".}
\end{aligned}
\tag{5}
$$

In particular, the signal will be set to hard if the expression $(SS \wedge RC \wedge GD \wedge ST)$ returns true which occurs when all the parameters have exceeded their respective threshold values. The other two levels, medium and soft, will trigger if three or two precursor respectively exceed their threshold values.

The creation of a volcanic alert system follows the seismic methodology, but with some important differences: (i) it is possible to estimate the threshold values for several geophysical parameters for which reference databases are available (volcanic tremor, concentration of magmatic gases, soil deformation, soil temperature). Through the acquisition of thermal infrared satellite imagery, carried out on a daily basis, it is possible to obtain information on the thermal contrast between contiguous areas of the volcano. This technique is of particular importance because, in correspondence with thermal anomalies, it is possible to evaluate the possible rise of magmatic bodies and identify potential areas where new vents may open; (ii) use the data relating to volcanic earthquakes to calculate the threshold value for all the seismic swarms that led to the opening of a volcanic mouth. In this way it is possible, similarly to the previous case, to define an alarm type

(hard, soft and medium) linked to the evaluation of Boolean expressions based on the threshold values of the five previously mentioned precursors.

2.4 Framework and implementation details

2.4.1 Software Architecture

The software architecture is based on Angular, a Single Page Application (SPA) that allows to efficiently manage the interactions between the components present inside the system, given by the following list:
(i) The main component (Component) i.e. the container of all the other components, centralizing all the main information. It stores the seismic and geophysical parameters (interacting with a proper database) previously computed on a statistical basis. The data are stored in arrays of objects made up of pairs [string, number] and triads [string, string, number] with the first and last elements consisting of particular keys. In the first object the first key indicates the seismic swarm code and the second one if the seismic swarm is open or closed (0 or 1). The last object consists of one triads with the first key indicating the swarm code, the second key if the seismic swarm is open, closed or inactive (0, 1 or -1) and the third key contains the average magnitude value of the seismic swarm. The intermediate objects are made up of pairs containing the swarm code and the magnitude value. Here is a practical example:

seismicSwarmDatabase = [
⋮ ⋮ ⋮
{swarm: "S4", state: 0} // beginning
{swarm: "S4", M: 2.5} // registration 1
{swarm: "S4", M: 2.8} // registration 2 (6)
⋮ ⋮ ⋮
{swarm: "S4", state: 1, $\bar{M}_m$: 2.7} // end
]

There are also a series of data structures in which the threshold values of seismic swarms and geophysical precursors and the corresponding standard deviations will be stored.

(ii) Component *GraphicTools* provides the service for formatting and displaying data, from the respective databases, in graphs and tables.

(iii) Component *UpdateData* periodically checks the Web pages relating to each of the components listed above for any updates of the published data. We briefly describe the activities that the *UpdateData* component performs regarding the updating of seismic data and soil temperature.

The *UpdateData* component establishes a connection with the web page that publishes the earthquake list, downloads the page to the local computer, parsers the page and extracts the updates that are stored in the container (main component) in an array of objects, named with the date of the current day.

The *searchSwarm()* method starts and searches for the most recent seismic events that have occurred in the past 24 hours, and transfers them to an array of objects. The last object of the swarm that closes the sequence is marked with 1 so that the array can expand in the presence of a new seismic sequence. At this point, the *calcAverageValue()* method checks whether the foreshocks of that sequence have ended with a mainshock.

If this happened in the third key of the last object it will be written 1, otherwise it will be written -1 to exclude the seismic swarm from the threshold magnitude (6).

The component for updating the soil temperature (*UpdateData*) periodically checks whether there is an update in the specific URL that will eventually be downloaded locally. The resulting image will then be divided into several areas and the temperature values calculated for each surface will be inserted into an array of objects. Each of them will contain a pair of keys (*Area, Temperature*) whose values correspond to the identifier of the area and the respective temperature value. Similarly to the previous case, at the end of the seismic swarm the array will be closed by inserting the value 1 in correspondence with the last object

of the sequence. Three specific methods, present in the main component, will calculate the necessary parameters (average, threshold and standard deviation) for the correct operation of the Early Warning System. Each of these values will be stored in the corresponding object arrays.

(iv) Component *DefineAlert*, by means of a set of Boolean expressions, compares the respective threshold values with the average values of seismic swarm sequences and geophysical parameters, establishing whether to launch a soft, medium or hard alarm, or avoids sending an alert. The *emitSignal()* method will take care of producing an acoustic-luminous signal if all the geophysical parameters present in the Boolean expressions assume a *true* value, that is, when they have exceeded their threshold values. The intensity of both signals will be proportional to the deviation between the threshold value and the current average value of the precursors:

$$|T_h - \bar{M}_d| \qquad (7)$$

The graphic button, present in the interface, will activate simultaneously with a beep signal through a typescript timer whose intensity will be increased or decreased according to the value assumed by the expression (5).

(v) A component that deals with the promulgation (*DispatchAlert*) of the alarm via email, sms, social networks (Facebook, Twitter), instant messaging (Telegram, WhatsApp).

2.4.2 Relationships and interactions between components

The class diagram in Figure 2.2 shows the structure of each component and its dependencies. In the *Component* class, the main container of the application, in addition to the seismic and eruption databases, there are several data structures such as *earthquakeThreshold* and *eruptionThreshold* consisting of objects of type {string, number, number} that respectively represent the level of magnitude, the threshold value and the relative standard deviation. The *main component* also consists of other methods, which take as input one of the arrays and through specific

algorithms calculate the necessary parameters (average, threshold and standard deviation) for the Early Warning System. The *addSwarm()* and *addData()* allow us to extend the database of seismic and volcanic precursors databases by adding new seismic swarms and experimental measurements by sensors and satellite.

The sequence diagram in Figure 2.3 shows the life cycle and the interactions between its components. The main *Component*, at specific time intervals for the different parameters, performs a call to the *UpdateData* component through the *parserSeismicData()* method and the other two related methods (*extractSeismicData()* and *searchSwarm()*): at the end of the execution, the control flow is returned to the main *Component* and thanks to the *data binding* the *UpdateData* component dynamically changes the fields of the *GraphicTools* component which refreshes the graphic updating it to new data. In an array of objects {string, string} there are web addresses through which it is possible to access updated data for all the geophysical components considered (seismic swarm and seismic and volcanic precursors).

The main *Component*, at the same time, uses the data extracted from the web page and through a series of methods nested within *calcThreshold()* calculates the main parameters required: the execution of the first method has only been reported in the sequence diagram to improve the readability of the diagram. At the end of the processing, the main *Component* performs a call to the *compareSeismicThreshold()* method of the *DefineAlert* component which deals with comparing the average values of the seismic swarm and the other recently measured parameters with the corresponding threshold values (with the associated standard deviations) to establish the alarm level. Subsequently, a call is made to *alarmLevel()*, a method nested in it, in which there are a series of Boolean expressions allow us to define the level of alert.

Finally, the *DispatchAlarm* component will take care of managing any alarm disclosure through the set of methods shown in the diagram (*sendMail()*, *sendSocialNetwork()*, ...).

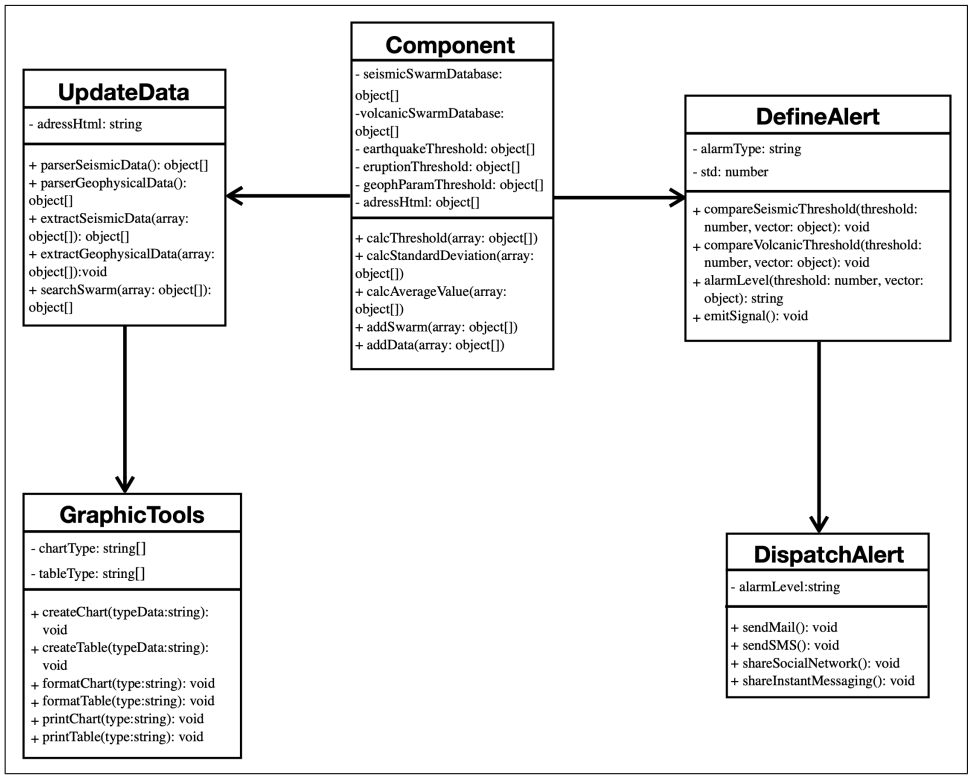


Figure 2.2: Class diagram

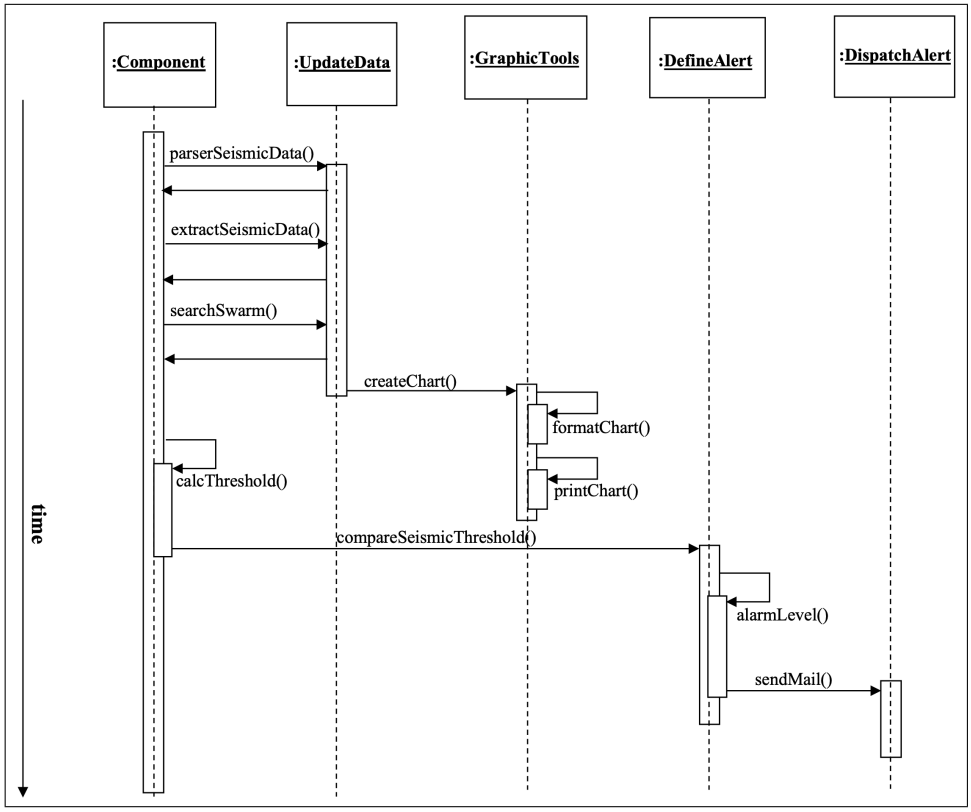


Figure 2.3: Sequence diagram

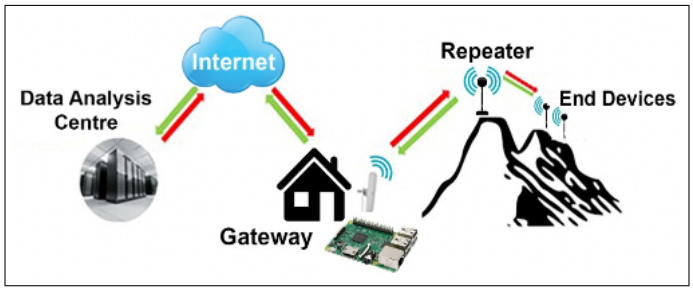


Figure 2.4: Repeater-Gateway transmission system [47]

2.5 Hardware and software architecture

2.5.1 Sensor networks

A fundamental prerequisite for the implementation of a Seismic Early Warning (SEW) is the presence, in the seismic territory, of a capillary network of seismometers and GPS sensors, recently replaced by GNSS (Global Navigation Satellite System). By this acronym we mean a constellation of satellites that, by sending a signal from space, allow specific receivers to determine their geographical coordinates (longitude, latitude and altitude) on any point on the earth's surface: any ground deformation, before, simultaneously or after a seismic event, will be highlighted by deviations from the original positions.

The test of the system, carried out in the Experiments section, was based on the available datasets, i.e. Seismic Swarms and Ground Deformation. Additional precursor parameters, in seismic areas where they are available, could significantly improve the results obtainable from SEW: concentration of Radon, CO_2, Arsenic and Iron, soil temperature are some of the many precursors that give significant anomalies before a destructive earthquake. The sensors network, arranged optimally for the seismogenic structures, must guarantee monitoring of the precursor parameters with measurements carried out continuously through a Repeater-Gateway transmission system, as in the case of ground deformation, earthquakes and soil temperature. For other precursors (Radon, Iron, CO_2 and Arsenic) the data acquisition can take place directly with on-site sampling. Figure

2.4 shows the acquisition-transmission scheme of the wireless network, consisting of three main components: the gateway, the repeater, and the end devices [47]. GNSS receivers, seismometers and geochemical sensors acquire the experimental data and send them to a repeater which amplifies the signal strength to be transmitted to the gateway, equipped with an internet connection, which routes them to the respective servers of the data processing center. And from this moment on, software agents come into play, carrying out a series of sub-activities to achieve the final goal corresponding to the definition of the current alert level.

2.5.2 Overall Structure of the Software System

After talking about the internal structure of the application, let's now consider the external one that we use the agents to carry out a series of operations concerning the transfer of data acquired from the sensors and all other activities related to the SEW.

The main features of the Multi-Agent System is based on some assumptions: (i) no agent can solve a problem on his own but must make use of the collaboration of the others to achieve the intended purpose; (ii) each agent differs from the others in the properties that distinguish it and the tasks it can perform; (iii) agents are divided and associated in a congregation, i.e. groupings of them that perform a series of semantically similar tasks.

With reference to the third point, we can consider that each group of agents acts in parallel and independently from the others, even if they share the same final objective. E.g., the cluster of agents SS (Seismic Swarm) acts in parallel with the clusters GD (Ground Deformation), RC (Radon Concentration) and ST (Soil Temperature): each group carries out similar activities to determine if there is an overlap between your current experimental data range and that of the corresponding threshold value, expressed by the relation (4). The interpretation of the data obtained from the n agent clusters and the definition of the alert level is the exclusive relevance of agent A. To verify that no malfunctions have occurred, a

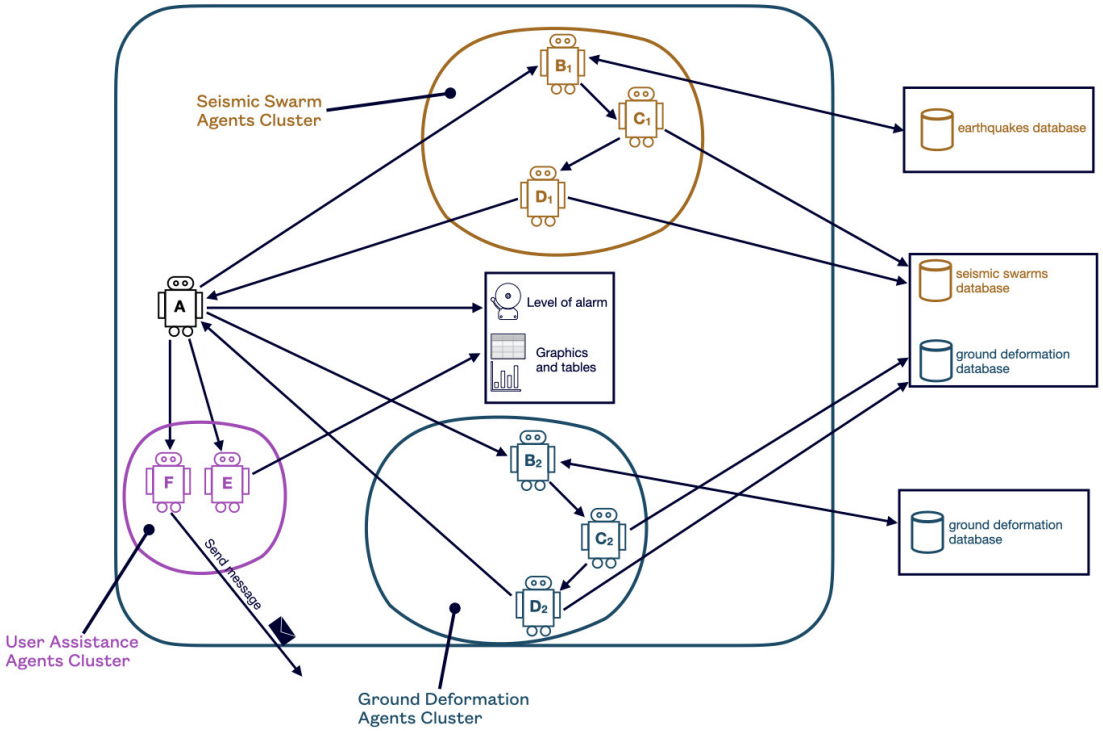


Figure 2.5: Agent interactions scheme

group of three demon agents (X_1, X_2 and X_3), periodically and alternately, checks whether the state of A is consistent, by sending it a message to which a response must follow. In case of no-confirmation, the role of the main agent will be assigned to one of the two "A substitutes" (A_1 or A_2) who will assume the same functions performed by A.

The detail of the interactions between the agents relating to different clusters is described in the following section.

2.5.3 Collaborative interaction between agents

The description of the interactions in the SEW system is based on the assumptions the MAS implementation concerns earthquakes with a magnitude greater than six and each agent is characterized by its internal state, that is, by variables and data structures which, at a given instant, contain specific values. Agents are server-side back-end components queried by the front-end. The system, still

under development, uses Java Agent Development Framework (JADE), a network-oriented framework that guarantees very efficient communication. The example shown refers to the geophysical parameter "Seismic Swarm", but the actions and operations carried out can also be considered substantially equivalent for the other geophysical parameters.

Collaboration and exchange of information can be summarized with the following activities, distributed over a series of agents: (i) download of experimental data from the corresponding servers where they have been stored by the sensor network; (ii) filtering according to certain rules that establish whether they are suitable for registration or not; (iii) analysis of current data and comparison with statistically calculated threshold values; (iv) establish if the alarm level must be updated defining its criticality; (v) formatting and display of data to be presented to the user; (vi) notification of a warning to a select group of scientists on any existing critical problems.

Figure 2.5 highlights the different roles assumed by agents B_1, C_1, D_1, belonging to the same group of agents, while A belongs to a hierarchically higher level. Every 10 minutes agent A sends a notification to agent B_1 that queries the internal server for the latest updates on seismic events occurred in that source area. In case of a positive response, it sends a message to agent C_1 which includes the magnitude (M_w), the hypocenter (H_p) and the date/hour (D) in which the seismic event occurred. Received the message, the agent C_1 compares the data received with those of the previous earthquake, stored in its internal state: the earthquake will be entered in the seismic swarm database only if it has $M_w \geq 1$ and occurred within 24 hours from the previous one, otherwise it is discarded. If the earthquake is inserted in the current seismic sequence, C_1 sends a notification signal to agent D_1 which activates and checks the earthquake frequency (F_E) in its own state in the last seven days, with the specifications defined previously (hypocenter, magnitude). If the frequency is sufficiently high (e.g. $F_E \geq 5$ earthquakes/day), D_1 calculates the average magnitude and the associated σ for the current seismic sequence and compares it with the corresponding threshold value. In case it reaches

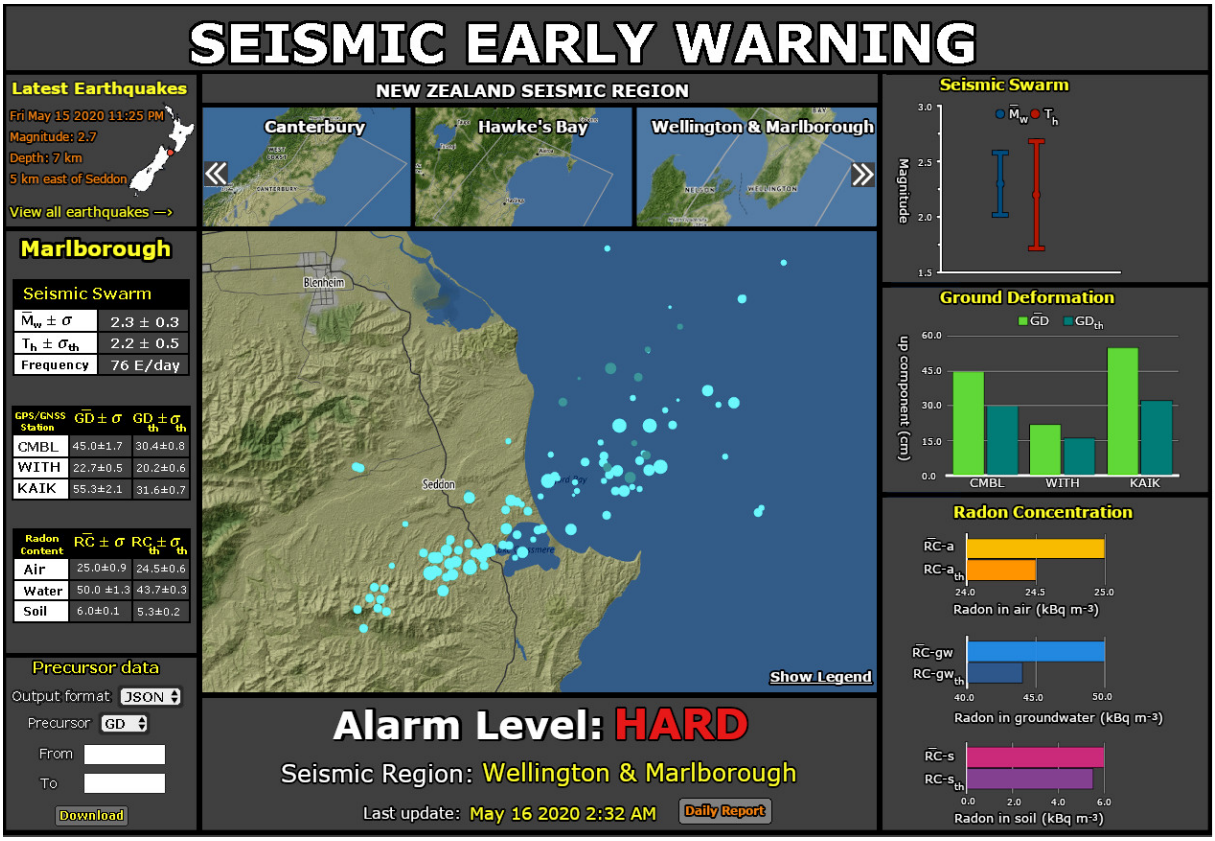


Figure 2.6: A mockup of the SEW dashboard

or exceeds the threshold value, D_1 updates the value of $\bar{M}_w$ in the database and sends a message to agent A, whose evaluation will take into account the frequency of the seismic swarm in the last days.

Next, based on the result obtained from the Boolean expression (5), it will decide whether to activate an alarming level and of which type (soft, medium or hard), sending a notification to the E and F agents, "specialized" in user assistance. In particular, the E agent will update the table and the respective graphs (histograms, box-and-whisker diagrams, etc.), while the F agent will send, via e-mail, a report to a small group of scientists. The document, created in an automated way, will report the experimental data that determined the activation of the specific alert level. At the end of the activity cycle, the clusters of agents listen for new notifications that can re-trigger the sequence of activities listed above.

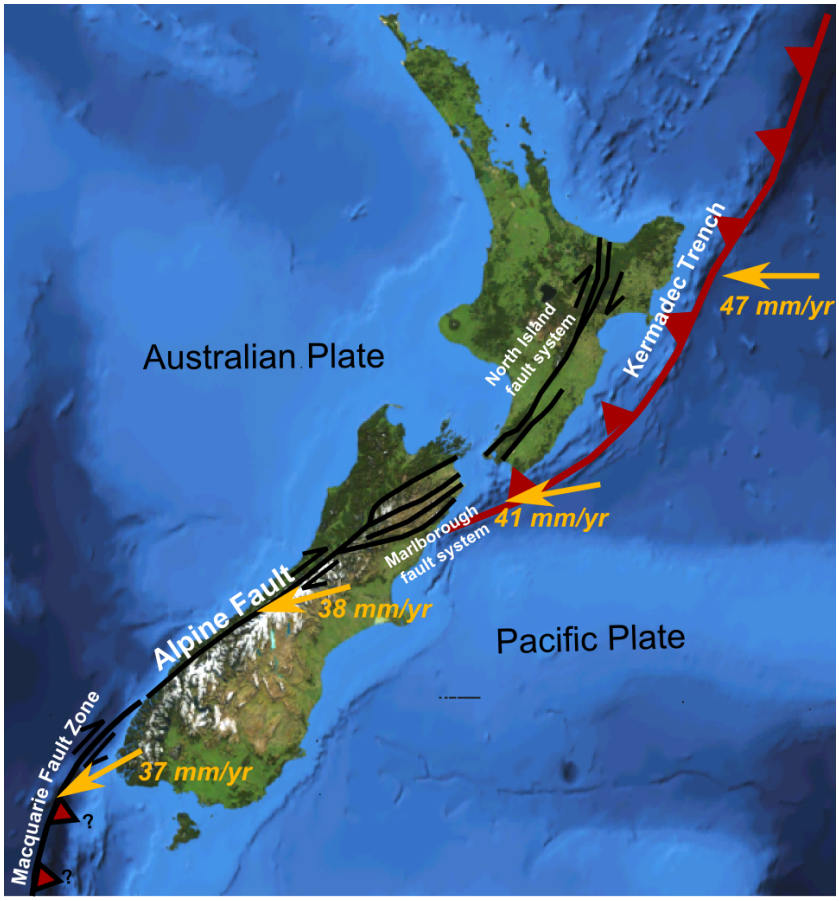


Figure 2.7: Scheme of the subduction area (Kermadec Trench) and the transform zone (Alpine fault) with the relative displacement speeds of the Pacific plate in collision with the Australian plate. (Credit: Mikenorton via Creative Commons https://commons.wikimedia.org/w/index.php?curid=10735284)

2.5.4 User assistance

Within the MAS, the purpose of Agents E and F is to assist users for the interpretation of experimental data and the notification of system status information documents. There are two degrees of access with level 2 users (scientists) who have more rights than level 1 (normal user). The main activities carried out by agent E can be summarized in: i) facilitating the interpretation of experimental data, showing them in real-time in the form of graphs and tables; ii) make them available in various formats, via download, for further research activities. At the end of the activity cycle, carried out by the various clusters of agents, which aim

at determining the alert level, A transmits the updates that have occurred: upon receipt of the notification, agent E is activated instantly by refreshing the SEW dashboard, which will show updated graphs and tables of each seismic precursor. A mockup of the SEW interface, currently under development, is shown in Figure 2.6. For both user levels, there is a button pointing to the precursor databases which contains recent and historical experimental data. Through a multiple-choice menu, it is possible to select one of the following possible formats: JSON, CSV and KML. When the user clicks on the "download" button, reactively and according to the selected choice, Agent E will take care of data extraction, formatting according to the selected format and starting the download process.

At the same time, Agent F creates a report in pdf to be sent via e-mail to a small group of scientists whose e-mail addresses have been stored. The document will be sent only in the presence of a hard level alarm and will present several standard fields: (i) the geographical coordinates of the area in which the seismic swarm occurred and the hypocentral depth; (ii) the frequency of earthquakes in the last two days; (iii) the average values and the relative standard deviation of the seismic precursors; (iv) further technical information on the instrumentation used, the seismogenic structure affected by the seismic swarm, etc. The information reported in the document have been extracted from the databases and system variables in which they are stored and assembled in a specific template, used for the realization of the report. In the case of the other alarm levels, no notification will be sent to the scientists, however it will always be possible to access an updated report, once a day, directly from the dashboard whose access is limited to level 2 users only.

2.6 Case study: New Zealand, a land with high seismic and volcanic risk

New Zealand is a region characterized by a high seismic and volcanic risk due to the presence of a fair number of active volcanoes and the particular geodynamic location, in the collision zone between the Australian Plate and Pacific Plates. For this reason, the area is covered by a dense network of sensors, some of which have only recently been operating, which allow continuous monitoring of different seismic and volcanic precursors. The data are made available to users through the GeoNet project (Geological hazard information for New Zealand) at the following URL: https://www.geonet.org.nz.

2.6.1 Seismotectonic overview

Within the GeoNet Quake Search section, New Zealand is divided into 10 seismic regions, from Auckland & Northland to Wellington & Marlborough. The intense tectonic and seismic activity is attributable to the presence of the Alpine fault, a large dextral transform structure, which crosses the southern part and marks the contact between the Pacific and the Australian plate. In the eastern off-shore area of the north island, the Pacific plate dips below the Australian plate: the phenomenon of subduction continues also at the Cook Strait and is the cause of deep earthquakes and the presence of active volcanism in the island of North. There are also a series of active secondary faults kinetically connected with the Alpine one, like Marlborough fault system, a set of four major faults which transfer displacement between Alpine fault and the Kermadec Trench (see Figure 2.7).

2.6.2 Experiments

The network of seismometers and GPS/GNSS sensors is well developed and represents a good way to test the seismic alert system. Each seismic region is covered

by a fair number of GPS/GNSS stations for the measurement of the ground deformation, even if for some stations the operativeness has occurred only in the last years and for others, the first registrations are from 1999. Of the three components that relate to displacement from the initial position (east, north, and up) only the up component was taken into consideration, relative to the vertical displacements of the ground. And this because the other two components, east and north, are mainly attributable to the displacement of the two plates.

The seismic data available on the "GeoNet Quake Search" page of the geonet web site were filtered by geographic coordinates, region and depth and downloaded in CSV format: the threshold value and the relative standard deviation were then calculated for two high magnitude seismic events. To download the data relating to the ground deformation, the GeoNet API was used, which allows the experimental data to be downloaded quickly, using special queries carried out in GET mode.

The SEW test was performed on the northern segment of the Marlborough fault system of the Wellington & Marlborough seismic region. The GPS/GNSS stations used for the calculation of the threshold values are those closest to the seismogenic structure analyzed, in which experimental data were available from 2004.

Seismic events occurred on 2013-07-21 and 2016-11-14 were considered, respectively of $M_w = 6.5$ and $M_w = 6.2$. Only two mainshocks have been considered, although they are made up of more than 600 seismic events in total, because catastrophic earthquakes of high magnitude, over the last twenty years, are quite limited in number.

For the ground deformation, the registrations made up to four months before the mainshock was considered and the threshold values for each of the three stations were obtained using the data relating to the two seismic events of 2013 and 2016.

In reality, by restricting the datasets to one month before the seismic event, the variation in the values obtained for the three stations is negligible and falls within the order of a tenth of a millimetre.

Seismic swarms	M_{ms}	$\bar{M}$	N_{se}
2013-07-21	6.5	2.4	340
2016-11-14	6.2	2.0	290
Threshold value: 2.2 **Standard deviation: 0.2** **Total number of seismic events: 630**			

Table 2.1: Seismic swarms before the mainshock on the Marlborough fault system

The *2013-08-16 earthquake* of $M_w = 6.5$ was used to test the correspondence between seismic swarms in progress and statistically calculated threshold values. A further test was performed on the *seismic swarm of May 2018* which as a final result did not give a mainshock.

2.6.3 Results

The data of the seismic swarms relating to earthquakes occurred on 2013-07-21 and 2016-11-14 are shown in Table 2.1: M_{ms} indicates the magnitude value of the mainshock. The threshold value obtained for the northern segment of the Marlborough fault system is of 2.2 ± 0.2. Table 2.2 shows the threshold values (T_h) and the respective standard deviations (σ), expressed in centimetres, relating to the 2013 and 2016 earthquakes for the three stations CMBL, WITH and KAIK. All stations are characterized by negative ground displacements which denote land subsidence before the mainshock.

The 2013-08-16 earthquake of $M_w = 6.5$, which occurred about a month later after the strong earthquake of July 2013, was used as a sequence to test the system.

Figure 2.6 shows one of the seismic swarms, in the Marlborough fault system, which preceded the mainshock: we can see the alignment of the hypocenters along a preferential direction that corresponds to the direction of development of the fault system that generated it (see Marlborough fault system of Figure 2.7).

GPS/GNSS Stations	$T_h(cm)$	$\sigma(cm)$	N_m
CMBL	-29	4.1	400
WITH	-4.6	0.8	400
KAIK	-17.2	0.4	400
Total number of measurements: 1200			

Table 2.2: Ground deformation before the mainshock on the Marlborough fault system

	Seismic Swarm	CMBL	WITH	KAIK
$\bar{GD}^*$	-	-24.6 $\pm$ 3.5	-4.9 $\pm$ 0.3	-16.8 $\pm$ 0.4
$\bar{GD}^{**}$	-	94.7 $\pm$ 0.5	9.4 $\pm$ 0.5	57.6 $\pm$ 0.5
$\bar{M}_w{}^*$	2.3 $\pm$ 0.4	-	-	-
$\bar{M}_w{}^{**}$	2.3 $\pm$ 0.7	-	-	-

Table 2.3: Test 2013-08-16 earthquake* and seismic swarm** on May 2018

Table 2.3 reports the average magnitude value and the relative standard deviation of the seismic swarm before the mainshock which is in the range of 2.3±0.4. The fields relating to the three ground deformation measuring stations show the values $\bar{GD} \pm \sigma_{gd}$. It can be seen that in all stations the intervals of the ground deformation in progress fall within the intervals of the threshold values $T_h \pm \sigma$. Hence, condition (4) is verified for both seismic precursors (SS and GD):

$$([\bar{M}_w \pm \sigma_w] \cap [SS_{Th} \pm \sigma_{th}] \neq \emptyset) \wedge ([\bar{GD} \pm \sigma_{gd}] \cap [GD_{Th} \pm \sigma_{th}] \neq \emptyset) \quad (6)$$

The evaluation of the Boolean expression for ground deformation corresponds to a logical AND between the three GNSS/GPS stations:

$$(CMBL) \wedge (WITH) \wedge (KAIK). \quad (7)$$

Figure 2.8 shows that in the three stations considered, before the event of August 2013, the ground deformation intervals intersects that of the respective

threshold values and therefore, according to the final result, the evaluation of the GD parameter returns true. A similar result is also obtained for the seismic swarm parameter with an almost complete overlap between the confidence interval in progress and that relating to the threshold value.

Table 2.3 also shows the results of the seismic swarm of May 2018 (about 115 foreshocks), indicated as two asterisks, which affected the same fault system. It can be observed that although the average value of the seismic swarm falls within the confidence interval of the respective threshold value, the expression (7) returns false because this does not happen for the ground deformation which has an inverse (positive) sign with respect to the corresponding (negative) threshold values.

2.7 Conclusions

With the integrated analysis, we aim to simultaneously analyze the experimental data of the physico-chemical precursors for which an adequate network of sensors is available. Acting in a complementary way means considering the results obtained by each parameter not disjoint from the others but which contribute, in different ways, to the achievement of the final objective. The innovation of the proposed model lies precisely in these short and simple concepts and the final evaluation of Boolean expressions made up of representative variables of each precursor allows each of them to make their contribution. In this way, it is possible to assess whether the transformation that a seismic territory is undergoing is on average attributable to those that occurred in the past in the periods preceding earthquakes of equal magnitude ($M_w \geq 6$).

According to the theory of dilatancy and asperity, the transformations that a territory undergoes before a strong seismic event produce ground deformations which, by fracturing, generates foreshock and catalyzes fluids from the surrounding areas, making their geochemical properties vary. If we consider that the entity of the deformations depends on the mechanical characteristics of the rocks present in each seismogenic area, we can consider that before each "characteristic earth-

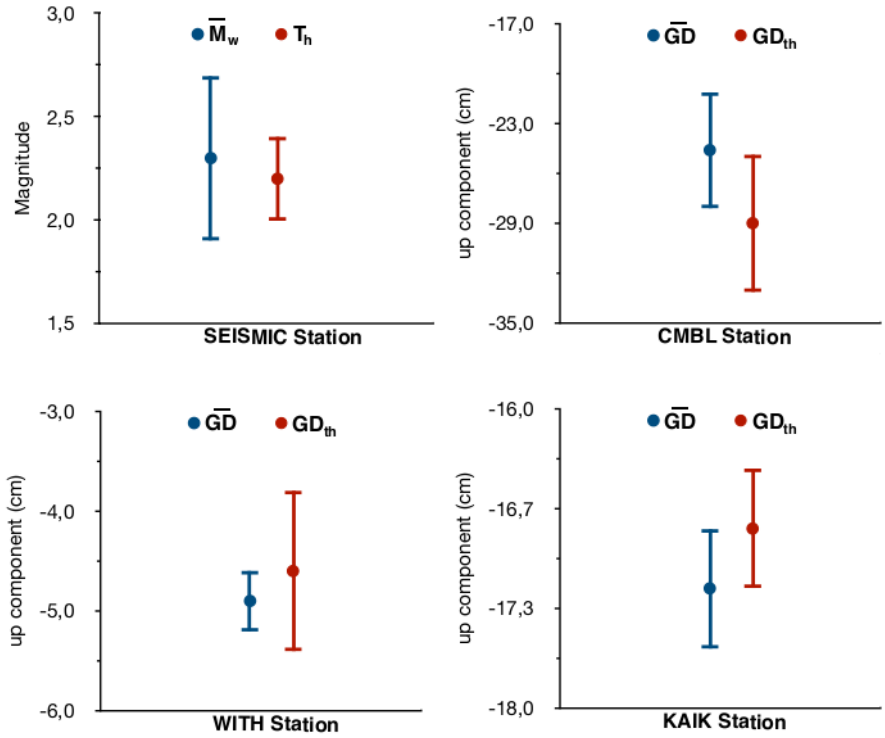


Figure 2.8: Comparison between the ranges of the data in progress and the threshold values for the ground deformation and the seismic swarms related to the August 2013 earthquake. Note that in all cases there is an overlap of the intervals.

quake" [45], physico-chemical anomalies, on average similar to those that occurred in the past, can be generated. The multi-agent structure greatly facilitates the process of acquiring and processing experimental data carried out in parallel by the agent clusters, each of which deals with a specific precursor. Two further agents, specialized in user assistance, take care of adequately formatting diagrams, tables and reports to be presented to users or sent to specific scientific groups to alert them of any critical states.

The results of the $M_w = 6.5$ earthquake test of 2013-08-16 on one of the seismic regions of New Zealand show an extensive overlap of the ranges $[\bar{M}_w \pm \sigma_w]$ and $[\bar{GD} \pm \sigma_{gd}]$ of both precursors with their respective confidence intervals of the threshold values and only small parts of the left interval are external to them. The ground deformation indicates that the areas surrounding the seismogenic structure undergo pronounced subsidence in the period before the seismic event. The

choice of this datasets is due to the possibility of using both seismic swarms and ground deformations starting from 2004, a combination not possible for surface earthquakes in the other seismic regions of New Zealand.

Conversely, the seismic swarm of May 2018, which affected the same area, shows that even if the seismic swarms in progress meet the threshold intervals, the deformation values in the three stations are largely outside the intervals $[GD_{Th} \pm \sigma_{th}]$: the absence of the mainshock is therefore in agreement with the result of the expression (7) which returns false. The results, therefore, confirm that a forecast based on a fair number of precursors can be a good solution for the implementation of a seismic alert system.

Chapter 3

Urban sprawl monitoring using image processing

3.1 Application Domain

The existence of green areas within urban centers is one of the main factors that improve the livability of cities. There are numerous benefits that the presence of vegetation cover produces both in the environmental and social fields: (i) aesthetic-ornamental function, regulation of the microclimate and expression of productivity [48]; (ii) influence on local air quality both directly and indirectly by altering the surrounding atmosphere and conditioning of local weather [49, 50]; (iii) the social, cultural and aesthetic functions offered by the open city spaces allow socialization, recreation and leisure in the open air [51]; (iv) open spaces favor social interactions between people, environmental education for children and promote a compatible relationship between humans and nature and consequently create urban sustainability [52].

The importance of green oases within cities had already been highlighted, in 1977, by the architect Christopher Alexander who had developed a series of design patterns [53], including "City Country Fingers", which involves the development of extensive prolongation of country land to the urban center. In several cities, especially in Japan, it is possible to recognize the imprint of urban development

based on country fingers. This term refers to extensive urban intersections of agricultural land or wooded hills which, from the peripheral areas, penetrate the city. Inside them, there are urban windows, called city fingers, whose development direction is opposite to those of the country fingers.

A parameter commonly used to identify vegetation, distinguishing the degree of vitality, is the NDVI (Normalized Difference Vegetation Index), an indicator of the greenness of the biomes [54]. This parameter is calculated using the satellite datasets belonging to the Red (0.630– 0.680 μm) and Near Infrared (0.845–0.885 μm) band, characterized by wavelengths that show a remarkable reflectance contrast. The principle according to which the vegetation tends to completely absorb the red and to almost completely reflect the wavelengths of the near infrared is the basis of the distinction between vegetation and anthropogenic constructions.

In order to quantify the contribution that green areas provide, it would be desired to evaluate the density of vegetation per unit of urban area. The quantitative analysis of urban green, expressed through some parameters such as area, density and length of the green bodies, has to be based on an automatic and correct labeling of the vegetation present within the urban center. This type of analysis is a valuable contribution in territorial planning studies and for the definition of policies to contrast the uncontrolled expansion of urban areas, defined as urban sprawl. The tools commonly used for the qualitative and quantitative assessment of vegetation cover in urban areas or entire regions are represented by GIS software systems which, through the analysis of the input satellite images, calculate the type of vegetation by dividing it into classes.

In such tools, however, user intervention is required. By means of the provided GUIs [55, 56], the user traces the vector limits of separation between the different themes. The geometric parameters' determination of the green bodies is also calculated manually by using measuring tools on the lengths and the areas. Then, SQL- based queries allow the user to extrapolate different characteristics from the context. Recurring examples are represented by the filtering of green spaces with an area below a certain threshold value, the display of neighborhoods characterized

by the highest and lowest density of vegetation, the filtering of urban areas closest to the major green areas, etc.

In this chapter we present an innovative approach that allows users to achieve the same results without the aid of GIS software systems and classic Landsat and Sentinel images or other paid suppliers of satellite images. Image processing techniques were used, implemented with the Python programming language and applied to high-resolution Google Earth images. The final goal is to demonstrate that a novel approach based solely on image processing techniques makes it possible to efficiently automate a series of activities, having execution times of the order of a few hundred seconds.

In addition to the classic activities of labeling, tracing the boundaries of separation of urban and green areas, calculation of areas, lengths and densities, we aim to implement queries that extract significant information from data stored in JSON files, with techniques similar to those of classic GIS applications. In addition to the city of Acireale the Python application was tested on the city of Kamakura, in Japan, one of the urban areas that shows a strong correspondence to the "City Country Fingers" design pattern.

3.2 Related Work

The calculation of the green areas within a territory is a prerequisite for several purposes, including the reduction of desertification risk and contrast urban sprawl. The quantification of the desertification risk in a given region is related to the comparison between the vegetation cover existing in different periods. An example is represented by the Churu district of Western Rajasthan, where long-term monitoring based on the NDVI time trend was carried out, which confirmed that some parts of the Churu district are subject to climate-induced desertification processes [57].

Some studies have highlighted a primary role, in the genesis of desertification phenomena, of land cover factors attributing a marginal weight to geomorpho-

logical factors [58]; while, in other cases, human activities, related to ecological recovery projects, represented the main causes [59].

The evaluation of green spaces within urban areas is also a topic of primary interest for environmental [48] and social [49, 50] reasons. Monitoring the relationship between urban and green areas during the phases of city expansion makes it possible to avoid an irrational development of urban centers, a phenomenon that is increasingly frequent in various countries of the world.

In recent decades there has been a renewed interest in urban sprawl and one of the reasons is the use of new technologies to measure the phenomenon based on digital cartography and georeferenced information [60]. The analysis of the vegetation cover present in a territory is preceded by the raster phase of cartographic labeling in which a classification is carried out by distinguishing green spaces from urban ones [61]. The latter is followed by a set of vector operations, generally carried out using GIS applications with a graphic interface, which can be summarized as follows: (i) tracing the boundaries between urban and green areas; (ii) calculating the main geometric parameters of each theme (area, width and length); (iii) SQL query execution [62, 63]. In some cases, urban green space analysis is performed using high-resolution images downloaded from Google Earth Pro, often used in conjunction with GIS applications [62, 64].

Software systems, such as GIS, assist the above activities but are manually carried out by users. The proposed approach overcomes such manual and cumbersome operations using automatic boundary recognition based on innovative image processing techniques applied to satellite images. Another technique that could achieve similar results is edge recognition. This technique is based on mathematical functions to identify areas of the image with an abrupt change in brightness. It has been applied in different images and for various purposes: (i) in face recognition for extracting edge maps from facial images under noisy [65]; (ii) for GIS-based detection of grain boundaries in deformed rocks from images of thin-section [66]; (iii) to detect edges from human being's X-Ray images based on Gaussian filter and statistical range [67]; (iv) for vehicle detection using edge-based candidate

generation and appearance-based classification [68]; (v) to identify the minutiae in the fingerprints, as minutiae matching is the widely used fingerprint detection and verification method [69].

In my initial work, an existing edge recognition algorithm was firstly adopted, however it did not produce encouraging results due to the application domain's peculiarity. The green areas' different gradations due to their typology and vigour degree and the presence of shadows due to the trees' foliage generate classification errors by identifying false urban areas within the green ones. Similarly, manufactured structures with reflectance values similar to vegetation create fictitious green areas within the urban perimeter. These factors alter the results by compromising the two themes' correct territorial labelling.

Artificial intelligence techniques based on Computer Vision have produced reliable results for various contexts: (i) Object Detection to identify all the entities in the image [70, 71]; (ii) image classification based on the content analysis by attributing a recognition label to the objects present in it (cat, car, people, etc.) [71, 72, 73, 74]; (iii) Face Recognition to identify the faces of people present in an image [71, 75, 76, 74]; (iv) Action Recognition to identify spatiotemporal relationships between entities within the image and to identify specific actions [77, 78, 79]; (v) Image Segmentation to analyze in detail the image previously divided into specific areas [80, 81, 82, 83].

Although Computer Vision algorithms can interpret each element of the image through advanced operations, the applicability of these techniques based on neural networks depends on the problem to be solved. A series of tests to be performed on a representative prototype of the image to be analyzed would make it possible to establish in advance the correspondence between the expected results and those obtained at the end of the processing.

Nevertheless, an approach based on neural networks requires a large dataset to reliably carry out the training phase, which, at present, is not available. The absence of similar applications developed in the past justifies the gap in datasets for labelling urban spaces. Furthermore, based on the previous considerations

expressed for the edge recognition technique, the presence of false positives within green and urban areas would significantly penalize this phase, even in the presence of significant datasets. The unsatisfactory results obtained in the initial phase of experimentation led us to consider a more robust deterministic approach, based on tuning the RGB values and capable of minimizing the critical issues otherwise encountered when using a Machine Learning approach.

3.3 Approach

The operational choices were based on a novel approach, using high-resolution images directly downloaded from Google Earth, with a spatial resolution of 2.5 m. This solution was deliberately used instead of the classic Landsat and/or Sentinel satellite images which, supported by GIS applications, would have facilitated the data analysis process. The choice of using a pure image processing approach, based solely on the PIL and OpenCV2 libraries of the Python programming language, has allowed us to implement complex techniques on RGB images. In this way, important geometrical parameters have been obtained with undoubted advantages on flexibility and versatility using digital techniques.

The first step is to download the RGB image, via Google Earth, e.g. of the Kamakura and Acireale city center, in high-resolution (2502 x 2607 pixels). From the first image a portion of 500 x 441 pixels has been cut, referring to some country fingers near the center of the cities. This let us simplify the analysis and the display of the results, and to reduce the execution time required for scanning the image pixels. A JSON file has been implemented for setting variables and processing resolution to enable or disable some tools and parameters calculation such as processing grid, area and length calculation, etc. The program, executed on the image mentioned above using the PIL library, lets us extract the RGB values of each pixel and perform a series of other operations described in the following.

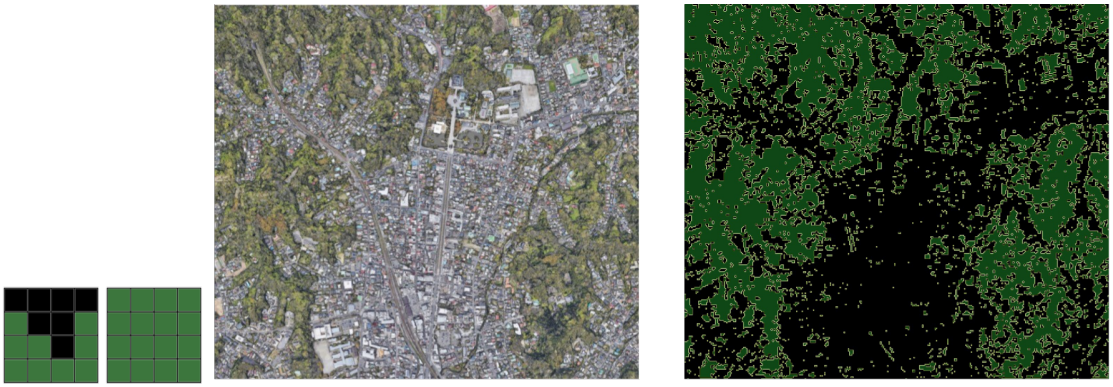


Figure 3.1: Labeling and boundary detection on a reduced prototype of Kamakura city (Maps Data: Google, SIO, NOAA, U.S. Navy, NGA,GEBCO).

3.4 Processing phases

3.4.1 Surfaces labelling and boundary tracing

The first operation was the labelling of the map, carried out on a prototype image to assess, first, the correctness of the choices adopted. The distinction between urban and green areas was made by means of an algorithm based on the get-pixel function (PIL library) which, for each pair of coordinates (x,y) returns the RGB values of the pixel. The threshold values of the three channels needed to discriminate against the two themes (urban area and green area) covered by the classification were then established.

The approach used was based on the acquisition of RGB values, for each of the two themes, in different parts of the image, calculating the average value for each of the three channels. We then proceeded to assign to each mean value a range of variation, then verifying, visually, the correspondence with the original image and adjusting the values in the case of divergence. The aim was to adopt a simplified approach limited to only two themes.

The mapping of the two themes was done on new files of the same size as the original one, loaded into the primary memory for the subsequent processing phases. Both files take on the meaning of layers, as they can be processed separately and merged into a single image or file, necessary to distinguish the boundaries between

the two themes. This last operation was carried out by the mergeImages function whose code is shown in Figure B of the Code Snippets appendix. The boundary that delimits urban areas from green ones is an element that belongs to both. For defining the two portions, i.e. green and urban, it is necessary to operate separately because in the same map they would be partially overlapped making it difficult to distinguish them.

At the end of this first processing phase, the green image layer will be obtained, characterized by the green color corresponding to the only mapped theme (vegetation). Instead the urban area will not be associated with any color (indicated with none) and will be displayed in black. Symmetrically the urban image layer, where the only mapped theme corresponding to the urban area is shown in light gray.

The mapping of the image was performed using a square tile of 4x4 pixel resolution. Considering that the spatial resolution is 2.5 meters, at each processing step an area of 100 m^2 is analyzed. During the analysis of each layer, to determine whether the specific theme or the absence of color is dominant within each mobile tile, the implemented calcPercent function deals with counting for each color the number of pixels within the tile and, if the percentage of one color is greater than 50%, the entire tile is colored with that specific dominant theme or otherwise with none color. The processing carried out by the mobile tile takes place for the entire image. Then, the pixels inside the tile are used to color the image by means of the PIL putpixel function. The path followed in the three years of PhD had the environment and the natural phenomena as a common denominator. Living in a healthy environment and minimizing the risks due to endogenous and exogenous phenomena are the main objectives of environmental monitoring. The surplus of collected data available at different spatial and temporal resolutions and their sharing lead to different analysis opportunities from past ones. These advantages are offered by satellite images which in recent years have proved to be a complementary tool to in situ analyzes of physico-chemical parameters to interpret the evolution of natural events. However, the result's quality depends

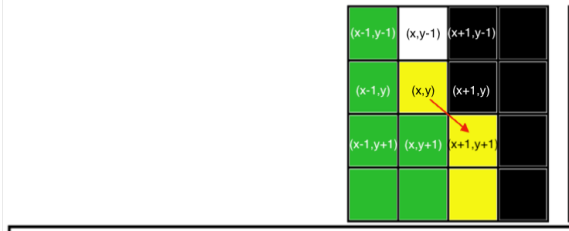
on the proposed solutions to describe the phenomenon correctly. An essential role concern the techniques used, which, in addition to extracting critical information, must also operate efficiently. Further restructuring phases have transformed from the original monolithic architecture to the microservices one, adopting a Kafka-based streaming platform for data exchange between containers. In some cases, it is essential to study the variations that occurred in a given time interval. The rate at which specific parameters vary can provide helpful information on the evolution of the phenomenon in progress and the degree of risk to which the territory in question is subject. These tasks are performed through the windowMobile function assisted by the built-in ones (imageLabeling, pixelCount, colorArea and calcPercent). The development of some of them is shown in Figure G.

The main steps of image labelling are shown in Figure 3.1. From the left: the mobile tile in the sampling phase; the tile at the end of the processing colored with the prevailing green theme; the detection boundary phase on the original image; and finally, the image after processing.

The next step was to trace the boundary in both layers. For this purpose, drawVerticalBoundary and drawHorizontalBoundary functions have been implemented to deal with the vertical and horizontal parts of boundaries respectively. The algorithm scans the entire image, i.e. the pixels per rows and columns. Every time there is a transition from green to black or vice versa, the intermediate pixel is colored yellow, both for horizontal and vertical scanning. The result will be a series of yellow lines that perfectly delimit the green areas from the black ones (Figure 3.1, right part). By executing the same procedure for the layer urban image we will also get the boundaries that separate the urban areas from the black ones, which represent those without a theme.

3.4.2 Boundary classification

The preliminary activity for classifying the boundaries was the red colouring of the image perimeter by acquiring the coordinates of each of these points. The next




```
countourCol="White"
while(actualCoordinate!=initialCoordinate) and (actualColor!=color["Red"])\
and (previousCoordinate!=actualCoordinate):
    previousCoordinate=actualCoordinate
    if img_1.getpixel((x+1,y+1))==color["Yellow"]:
        while img_1.getpixel((x+1,y+1))==color["Yellow"] and (actualCoordinate!=initialCoordinate)\
        and (actualColor!=color["Red"]):
            y+=1
            actualCoordinate=(x,y)
            actualColor=img_1.getpixel(actualCoordinate)
            lenghtBoundary+=1
            index+=1
            img_1.putpixel(actualCoordinate,color[countourCol])
            copyCoordAsJSON(boundaryCoord,actualCoordinate,number,index)
    elif img_1.getpixel((x,y+1))==color["Yellow"]:
        while img_1.getpixel((x,y+1))==color["Yellow"] and (actualCoordinate!=initialCoordinate)\
        and (actualColor!=color["Red"]):
            x+=1
            actualCoordinate=(x,y)
            actualColor=img_1.getpixel(actualCoordinate)
            lenghtBoundary+=1
            index+=1
            img_1.putpixel(actualCoordinate,color[countourCol])
            copyCoordAsJSON(boundaryCoord,actualCoordinate,number,index)
```

Figure 3.2: Description of the followBoundary algorithm.

step has been performed by defineGreenUrbanUnits and followBoundary functions (see Code Snippets Figure A and Figure D). The first scans the entire image and checks the pixel color. If the color is the same as the boundary, the pixel is colored white and the coordinate (x,y) is inserted in the file coordinates.json. The color changing is essential to prevent it from retracing the same path during processing. Next, the followBoundary function is executed, consisting of a main while loop with a series of additional loops nested inside. The main cycle gives the exit condition that will occur when one of the three conditions is true: (i) if the initial coordinate is equal to the final coordinate, which occurs when the boundary closes; (ii) when the current color is red, that is, the coordinate corresponds to that of a border; (iii) when the initial coordinate corresponds to the previous one.

The while cycles nested within the main one check each of the eight positions around the current pixel: if in one of these positions there is a pixel of the same color of the boundary (yellow), control will move to it and its coordinates will be

inserted in the JSON file. E.g., if the current coordinate is (x,y) and the check is for the pixel (x,y+1), then until a yellow pixel is in it the cycle will be repeated. Figure 3.2 shows, in yellow, the boundary that separates the vegetation (in green) from the absence of color (in black); in white the visited boundary. The left image shows that coordinate (x+1,y+1) is recognized as belonging to the boundary. The right image shows that the loop of the previous position exits and the while relative to the coordinate (x,y+1) is entered, which writes the coordinate to the JSON file and colors the coordinate in white. The corresponding code is also shown.

The additional exit conditions from the secondary while will be equal to the first two main cycle conditions. Ultimately, the eight secondary while cycles check all possible positions where the boundary may be at each iteration. The white coloring of each boundary pixel allowed for immediate feedback on the correctness of the operation, i.e. the algorithm reliability is proven if the boundary changes from yellow to white. During the execution of the algorithm, a JSON file was generated, called boundary_parameters.json, reporting for each boundary: (i) type of area, distinguished in "Urban Area" or "Green Area"; (ii) area value, in pixels and square meters; (iii) boundary length, in pixels and meters; (iv) length and width of the unit, in pixels and meters; (v) the pair of coordinates (x,y) of each boundary point; (vi) initial boundary coordinate; (vii) final boundary coordinate corresponding to the last pixel before closing.

Once the points of each boundary have been obtained, and given a distinct number in the JSON files, the geometrical parameters of the areas, have been calculated.

3.4.3 Parameterization of green and urban areas

After distinguishing green areas from urban ones during the first labelling phase, the next step concerns calculating the main geometric parameters. We estimate the length and width of the boundaries and the area enclosed within them. A series of algorithms characterized by different computational complexity have carried out

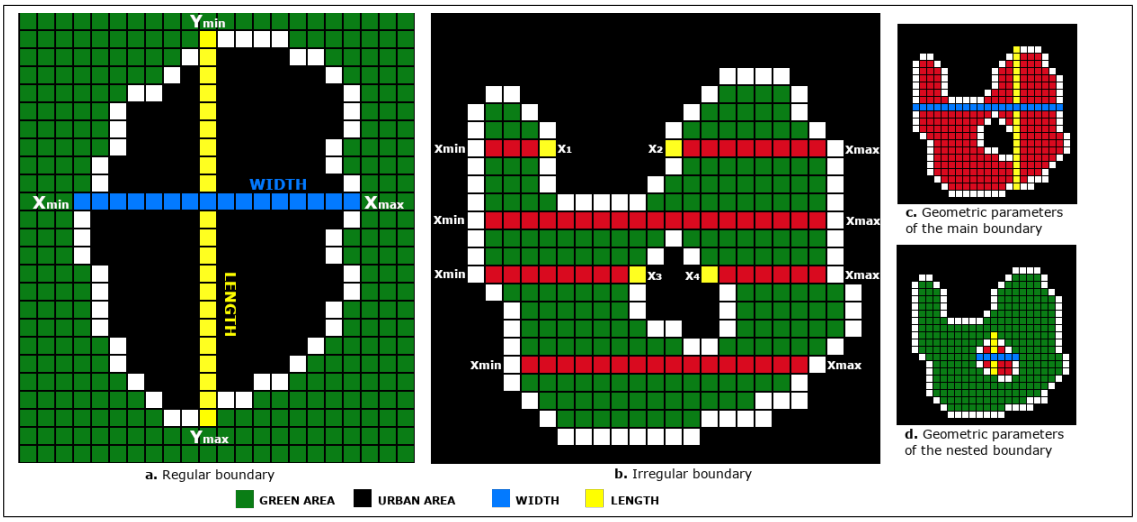


Figure 3.3: Some techniques used to calculate the length, width, and area of green and urban bodies. Each small square represents a pixel, and its size has been intentionally exaggerated

these tasks. Table 3.1 shows the various functions and their execution times. The calcXMaxOrMin and calcYMaxOrMin algorithms have the lowest performance at the resolution of 2502 x 2607 pixels.

Let us now examine the characteristics of each algorithm. The calcXMaxOrMin function for each y coordinate value of the boundary points calculates the minimum and maximum value of x. It stores them in a particular field of the JSON file. This procedure is necessary because, in irregularly shaped boundaries, the limit can present some curvatures, interrupting the surface continuity. In this case, it occurs that for every y, there are several values of x, and only two of them will be the extreme points of boundary. Figure F shows a code snapshot of the calcXMaxOrMin function.

The central image of Figure 3.3 shows a green area contained within an urban agglomeration (in black). In the upper part, the boundary colored in white has a winding morphology and tends to fold inside the surface. It can see that X_{min} and X_{max} represent the horizontal extremes of the limit for each value of y. In yellow, the pair of abscissas (x_1, x_2) although they have the same y value as the corresponding X_{min} and X_{max} values, represent internal points due to the indenta-

tion of the boundary. Also, the pair (x_3, x_4) contained in the interval $[X_{min}, X_{max}]$ corresponds to the abscissas of the boundary that delimits an urban area nested inside the green body. It is possible to make similar considerations for the Y_{min} and Y_{max} values, representing the vertical boundary extremes for each x. To obtain the length and width of each boundary is necessary to calculate $X_{max} - X_{min}$ and $Y_{max} - Y_{min}$ length segments for each y and x of the limit. The estimate of the two dimensions will correspond to their maximum values:

$$Width_i = max_i(X_{max} - X_{min}) \ \forall y \in B$$

$$Length_i = max_i(Y_{max} - Y_{min}) \ \forall x \in B$$

where B corresponds to the set of points of the i_{th} boundary.

In images 3.3a-c, the length and width are shown respectively for regular and irregular shape boundary. Figure 3.3d shows the two dimensions for the urban body contained within the green one calculated through the calcWidthAndLength function: a code snapshot is shown in the Figure C.

To calculate the area subtended by each green or urban body must be counted the pixels between the pairs of values (X_{min}, X_{max}) for each y of the boundary. The areas nested within the main one will be excluded from the count during the horizontal lines scanning. Repeating the same procedure for each horizontal line obtains the total number of pixels inside the urban or green body. Multiplying this value by the area of each pixel, i.e. spatial resolution raised to the second power, we will obtain the surface subtended by that specific boundary. The pseudocode of the Calc Area algorithm is shown in the Figure E.

Figure 3.3b illustrates the pixels count, highlighted in red, between the extreme values X_{min} and X_{max}. Note that the coloring stops for those present in the interval $[x_1, x_2]$ due to the boundary indentation. The same applies to the pixels between the x_3 and x_4 values, belonging to the urban area nested within the green one. Figure 3.3c shows the count conclusion of the internal pixels, entirely colored in red, whose value will correspond to the area of the green body. Similarly, figure 3.3d illustrates the count performed in the urban area included within the main boundary.

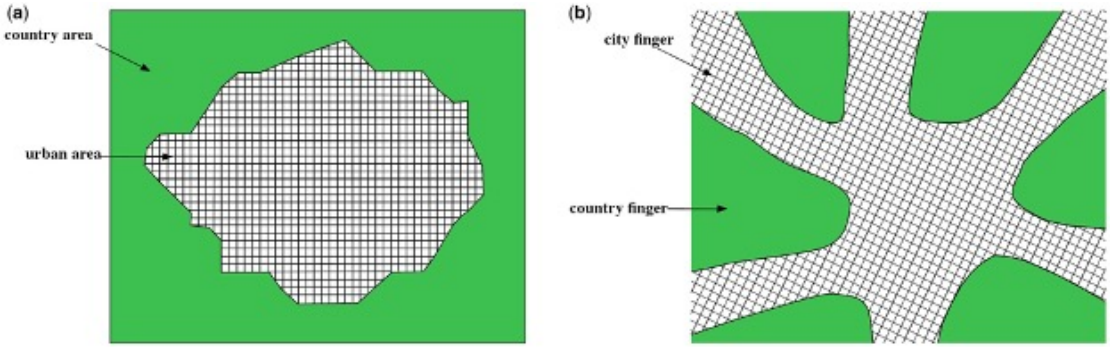


Figure 3.4: Two examples of classification of urban areas relating to the 'City Country Fingers' pattern. In case (a), there is no adherence to the pattern: the agricultural lands surround the city and have no intersections with it. In (b), there is maximum adherence to the pattern (ideal case): fingers of the countryside branch off from all four cardinal points and penetrate the inner city

3.5 Experiments

3.5.1 Green spaces in the areas of Kamakura and Acireale

Classifying urban areas by considering intersections with country spaces from peripheral areas can be considered an efficient method to define the livability of a medium-large city. Having access to rural areas intercalated within cities allows us to improve the quality of life of the inhabitants: one of the main advantages is the possibility of the frequent contact with nature without having to undertake prolonged and stressful journeys when they are in remote areas.

In several cities, especially in Japan, it is possible to recognize the imprint of urban development based on country fingers. This term refers to extensive urban intersections of agricultural land or wooded hills which, from the peripheral areas, penetrate the city. Inside them, there are urban windows, called city fingers, whose development direction is opposite to those of the country fingers.

The concept introduced by Christopher Alexander in 1977, on which the 'City Country Fingers' pattern was developed, was based on an ideal situation with the urban area intersected by country fingers from all directions and prograding toward the city center (Figure 3.4-b).

Algorithms	Execution Time (sec)	Resolution (pixel)
windowMobile	18.66	2502 × 2607
defineGreenUrbanUnits	11.06	2502 × 2607
calcWidthAndLength	0.37	2502 × 2607
calcArea	23.11	2502 × 2607
calcXMaxOrMin	31.77	2502 × 2607
calcYMaxOrMin	30.97	2502 × 2607
mergeImages	5.29	2502 × 2607
Whole application	320.69	2502 × 2607
Whole application	693.76	3753 × 3911

Table 3.1: Execution times of some algorithms of the application. Performance for the entire application, at different resolutions, is shown in the last two rows of the table.

To recognize and analyze, in an automated way, these particular structures, a Python-based application was created. Starting from the original high-resolution image of Google Earth, a complete analysis was performed, labeling and delimiting urban and vegetational areas and extrapolating the main geometric parameters of the country and city fingers.

In the analyzed cities, two main morphologies relating to the country fingers consisting of wooded reliefs and agricultural land were recognized. The first frequently present a three-dimensional development with fractal-like geometry that recursively tends to reproduce itself. In this way, various branches of the main country finger are generated, within which smaller city fingers, organized hierarchically, develop (Figure 3.5 below). The latter has a planar development and a linear structure with a hierarchical organization of the country and city fingers practically absent (Figure 3.5 above).

3.5.2 Results

The functions implementation of the table 3.1 is reported in the Code Snippets section. The results show a good agreement between the original map and the one obtained at the end of the processing. This correspondence concerns the boundaries tracing, the labelling of the areas, and the geometric parameter's consistency extracted for each area and boundary. The images in figure 3.5 confirm the veracity of the previous statements.

The execution times for several algorithms and for the entire application are shown in Table 3.1, and the last row shows performance when increasing the resolution. It shows a non-linear variation of the execution time. For the larger image, calcWidthAndLength function performs the best as the processing simply uses the coordinate pairs of the boundaries. As expected, calcXMaxOrMin and calcYMaxOrMin functions have increased execution times.

3.5.3 Filtering through JSON query

The last step of the geospatial analysis concerns the queries execution, applied on data from the previous phases, stored in a JSON file. Several queries based on JSON files have been implemented, which contain all the boundaries coordinates and the maximum and minimum values (xmin, xmax, ymin, ymax) for each different y and x. We report and analyze one of the most significant: filtering out the green areas of small size which, in relation to the need to highlight the "City Country Fingers" pattern, can represent e.g. an area belonging to some family, such as a garden, hence not a part of a finger, etc. They use techniques described above to select and colour areas and boundaries, highlight green and urban bodies with specific values or ranges of surface, length, or width, and so on. Figure 3.6 shows the results of the initial processing phase and those of some queries. Images 3.6a and 3.6b show the original satellite imagery and the labelling results of green and urban areas (in black). Image 3.6c views the query result selecting green and urban areas with a surface between 100 and 400 pixels. The two themes

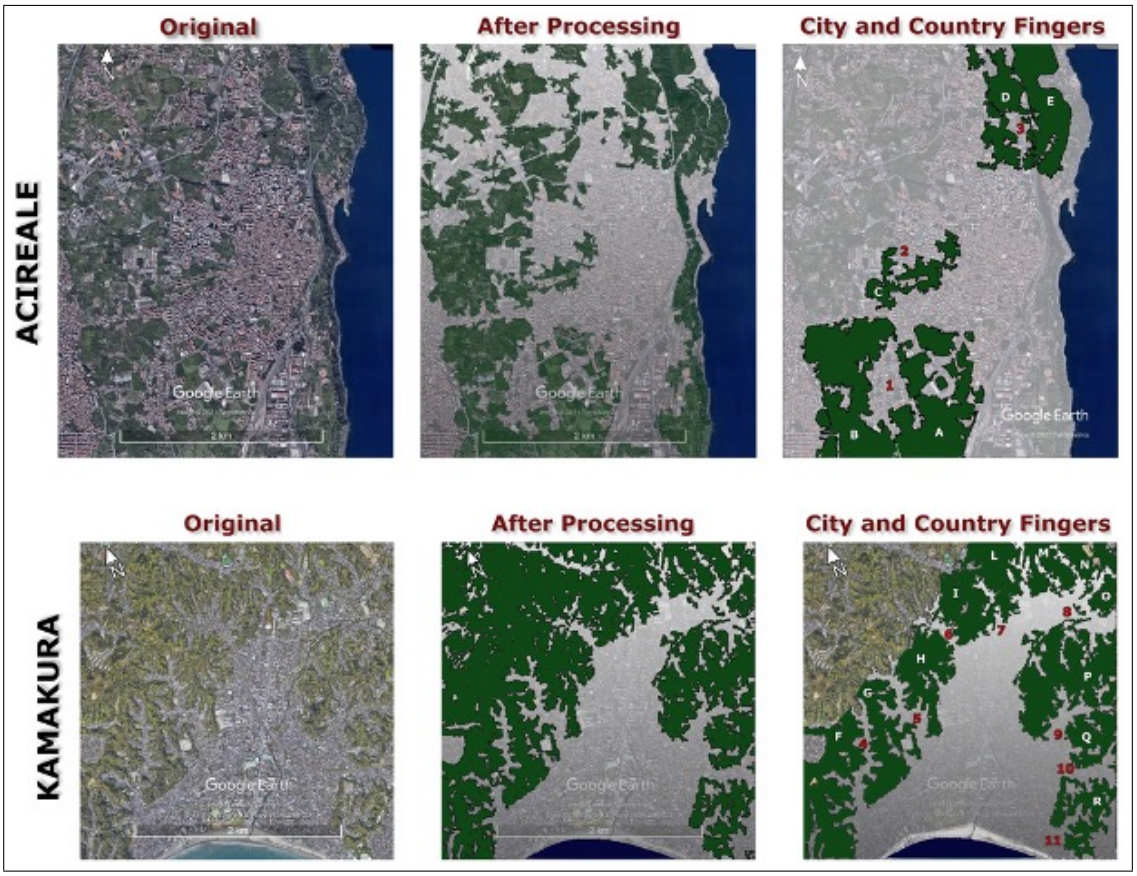


Figure 3.5: The results of the image processing of the Acireale and Kamakura cities. In the figures on the left the original image, in the central ones the result after the processing phase and in the figures on the right of the country and city fingers for both places indicated by letters and numbers. The vegetation is shown in green and the urban center in transparent gray

are colored, respectively, with blue and red colors. The last query (image 3.6d) removes the green and urban areas of small size that can represent a noise for certain types of analysis. The coloring of the entire area and the boundary, with the same color black as the urban area, will be carried out by means of the putpixel function. Additional queries have been implemented. These are just some of the many information that can be extracted, by performing queries.

Figure 3.7 shows a snapshot of the *query_1* executed on the JSON file: after loading the *boundary_coord* (list of coordinates) file, areas that are less than 200 pixels are removed. The number of iterations will depend on the size of xmax (or xmin) of the selected boundary. The coloring of the entire area and the boundary,

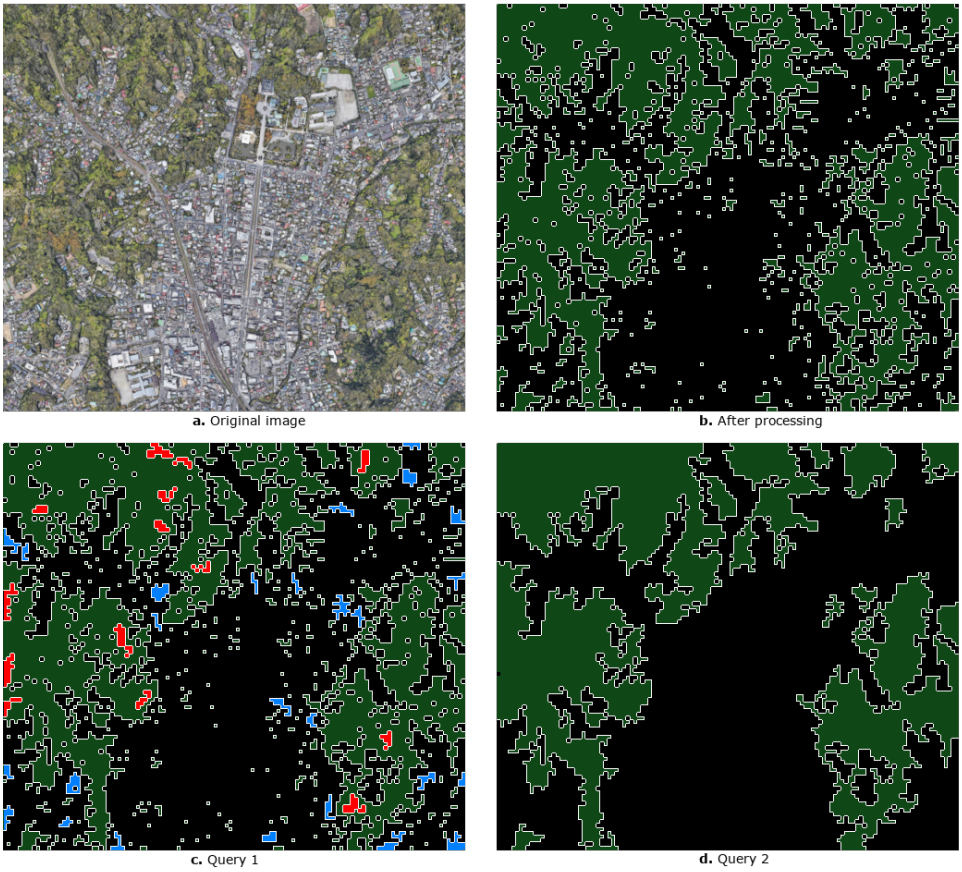


Figure 3.6: Some queries in JSON format. a. Original Google Earth image with a spatial resolution of 2.5 meters (Maps Data: Google, SIO, NOAA, U.S. Navy, NGA, GEBCO); b. Labelling of the urban and green areas; c. Query 1: highlighting of the urban and green areas with a surface between 100 and 400 pixels; d. Query 2: removing noise from the image b.

with the same color gray as the urban area, will be carried out by means of the putpixel function. Additional queries have been implemented. Specifically, *query_2* returns the total surfaces covered by green areas and urban areas through the union of the respective individual areas. Finally, *query_3* (not present in figure 3.7) colours in red all the green areas between 200 m^2 and 400 m^2. These are just some of the many data that can be extracted, by performing queries, from the final image obtained by merging the two layers.

```
def jsonQuery(img,boundary_coord,color):
  query_1=True
  query_2=False
  for j in range(1,len(boundary_coord)):
     if query_1:
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Green_Area'\
        and boundary_coord[f"Boundary_{j}"]["Area"]["pixel"]<200:
            #area
            for i in range(1,len(boundary_coord[f"Boundary_{j}"]["X_max"])):
                x1= boundary_coord[f"Boundary_{j}"]["X_min"][f"x_{i}"]
                x2= boundary_coord[f"Boundary_{j}"]["X_max"][f"x_{i}"]
                img["main_image"].putpixel((x1,x2), color["Gray"])
            #boundary
            for k in range(1,len(boundary_coord[f"Boundary_{j}"]["Points"])):
                (x,y)=boundary_coord[f"Boundary_{j}"]["Points"][f"p_{k}"]
                img["main_image"].putpixel((x,y), color["Gray"])
     if query_2:
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Green_Area':
            area["Total_Green_Area"]+= boundary_coord[f"Boundary_{j}"]["Area"]["m^2"]
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Urban_Area':
            area["Total_Urban_Area"]+= boundary_coord[f"Boundary_{j}"]["Area"]["m^2"]
```

Figure 3.7: A snapshot of some JSON queries

3.6 Conclusions

An approach, and a corresponding Python application, was proposed to automate many operations usually performed manually by using a GIS. The tracing of the boundaries that separate classes of different pixels and the calculation of the main geometric parameters of the different thematic units are the main operations implemented. The semantics of the proposed algorithms have an innovative character. The boundary detection between different types of terrain or between urban areas and vegetation, and the calculation of lengths and areas has required, up to now, the direct intervention of the user. Applications based on three-dimensional satellite images, or the most important GIS software such as ArcGIS or QGIS, have specific tools to draw boundaries and calculate areas. In irregularly shaped surfaces, the use of such manual tools has obvious limits in the correct evaluation of the contours and areas. The proposed techniques let us improve the reliability of the results obtained in the presence of irregular shapes.

In the initial step of software implementation, a comparison was made with standard techniques (edge recognition and machine learning), carrying out some tests based on the algorithms available in the literature. The attempts were aimed

at evaluating their correspondence for the application. However, the results obtained on a prototype of the small-size image (500x400 pixels) showed unsatisfactory results. The characteristics of both approaches do not meet the analysis requirements needed by the application. The extreme variability of the house colours that characterize an urban area shows clear limits in applying the edge detection technique or the neural network-based approach. Green areas in which there is a transition from zones of greater vigour to those characterized by dry vegetation, shaded areas within green spaces or green buildings or infrastructures are all factors that can lead to false positives, i.e. recognition of urban surfaces instead of green ones or vice versa. These problems of the GIS application's context lead to final results that differ from the pre-established ones based on a binary classification aimed at quantifying green and urban areas.

The GIS application has been tested on high-resolution Google Earth images. However, simple tuning can also be adapted to types of images with different formats, resolutions and types of input data.

In conclusion, the obtained results confirmed the reliability of the proposed approach. They are a contribution of new ideas and an example of the development of algorithms capable of abstracting the application from the subjectivity of the freehand techniques used by the user.

Chapter 4

Image swarm processing for the evolution of natural phenomena

4.1 Introduction

Analyzing a set of images relating to a specific part of the territory in a given time interval can represent a powerful technique for different purposes. In various sectors, it is often necessary to know the variations a given phenomenon has in space and time. A privileged field for this type of analysis is the scientific one.

Monitoring natural phenomena is crucial to understanding their short and medium-term behaviour. For this purpose, an application has been implemented which can autonomously detect variations in the same area and in one or more pre-established time intervals. There are two fundamental prerequisites for this analysis: (i) the variations must be assessed quantitatively; (ii) the processing to obtain the gradient maps must be done efficiently.

In the first case, determining the main geometric parameters is fundamental for correctly interpreting the phenomenon's evolution. For example, if we want to get the thermal gradient map of a volcanic area, we need to calculate the areas, boundaries, length and width of each heat cell. Optimizing the performance of the image swarm processing phases is a further factor that can affect its reliability, that is, the system's ability to behave according to the previously established

specifications.

Before the execution to identify the real variations of the swarm of images, several simulations were performed on multiple images relating to the two Japanese cities of Kamakura and Ōhata. Boundary labelling, area calculation and some queries proved to be the most critical and costly steps. The maps obtained from the partial and total processing of the swarm made it possible to get helpful information relating to the phenomenon of uncontrolled urban sprawl.

4.2 Related Work

The swarm analysis of satellite images has been used in the scientific field to extract meaningful information. Frequently, in the massive processing of satellite images, a strong parallelization of some components of the application is chosen to increase its performance. Multiple image processing has been related to different contexts and purposes: (i) using an automated workflow to extract information related to smallholder farming [84]; (ii) implementing a fully automatic image processing chain that carries out all processing steps by switching from sensor-corrected optical images (level 1) to web-delivered map-ready images and products without operator's intervention. This automated workflow aims to respond to the growing need for automatic and fast processing of satellite images [85]; (iii) developing a new approach to distributed processing of considerable amounts of satellite images, using HIPI as an alternative to manipulating them. Adding the new tiff format to the HIPI framework helps preserve information and massively process and analyze satellite images for faster results than traditional remote sensing [86]; (iv) building a high-performance system for processing a large daily volume of Chinese satellite images (approximately 1,500 scenes or 1 TB per day) in a timely manner and generating geometrically accurate orthorectified products [87].

These approaches based on massive processing aim to carry out their activities faster than those performed on single images [85, 86, 87, 88]. In addition to paying particular attention to performance, our proposal has a highly innovative

implication compared to previous approaches based on image swarms. Interpreting the evolution of some natural phenomena through their variations in time and space is the cardinal point on which the primary interest of the application is focused. The contribution provided by the complementary analysis of different time versions of the same place makes it possible to extrapolate the changes that occurred within a specific time interval. At the same time, the spatial analysis makes it possible to highlight the changes undergone by portions of the surface that outline the spatial evolution with which the phenomenon tends to spread in distinct parts of the territory.

The innovative nature of the research is related to the absence of previous works that use the massive processing of satellite images as a key to interpreting natural phenomena to trace their space-time evolution.

4.3 Environmental monitoring techniques

The environmental monitoring can be carried out directly or indirectly. Direct monitoring is carried out with in situ measurements by means of sensors that allow the detection of specific environmental parameters. The type of such devices depends on the natural phenomenon to be monitored. Thus, in the seismic and volcanic field, sensors are used to measure earthquakes [89], ground deformation [90], ground temperature [91, 92], etc. Generally the sensors are placed in stations of measurements located in different areas of the volcano or fault area, using in some cases an IOT-based transceiver plugged onto the development board and an embedded 3-d axis accelerometer [93].

Other types of sensor-based environmental monitoring is represented by measurements to control the quality of water [94, 95] and air [96, 97], essential factors for improving the quality of life in cities and contrasting of pollution which may lead to the onset of disease in the resident populations. Further forms of monitoring of physical parameters concern electromagnetic [98, 99] and noise [100, 101] pollution, which involve similar health risks.

Natural phenomena can be controlled in an alternative way than monitoring physical-chemical parameters performed directly on site. Depending on the case, the analysis is applied to satellite images with higher or lower resolution. This way, GUI or image processing techniques evaluate the variations generated by a specific natural phenomenon in a time interval and a given area. The variations are obtained by measuring lengths and areas using special graphic tools in the GIS software. This analysis allows us to understand the velocity with which the natural phenomenon evolves and the extent of the changes undergone by the different portions of the territory over time. This technique is applied in various fields in the scientific world: (i) unsupervised monitoring vegetation [102]; (ii) retrieval of land surface temperature [103, 104]; (iii) prediction of upwelling events in the coastal areas [105]; (iv) monitoring spatio-temporal changes of terrestrial ecosystem [106]; (v) environmental monitoring of the city [107]; (vi) environmental monitoring of submarine volcanoes [108].

One of the advantages of the satellite interpretation method is economical [109, 110]. The data analysis requires only image availability, which, based on the phenomenon, can also be performed on lower-resolution representations. Otherwise, carrying out the in situ detection of physical and geophysical parameters requires chemical analysis, reagents and vast sensor networks, which need considerable costs and time [111, 112].

In some cases, the two types of surveys are performed jointly: the punctual and discrete measurements performed on-site are integrated with the evaluations deduced from the interpretation of the aerial images. This dual approach makes it possible to constrain better the results obtained from applying a single method [113, 114, 115].

4.4 Controlled Experiment: urban sprawl in the Ōhata and Kamakura cities

Experiments were conducted to test the method's reliability, which, through elaborating a set of images, aims to acquire helpful information to interpret the evolution of some natural phenomena.

The Ōhata and Kamakura Japanese cities were chosen for their conformation of the urban greenery, according to the "City Country Fingers" pattern illustrated in the previous chapter (Figure 3.4). The complexity of the boundaries between green and urban areas due to frequent intersections is a proper testbed for the application.

The method illustrated in the next section is based on the simulation of the green and urban variations that occurred in a specific time interval and a given area. The fundamental purpose of this preliminary elaboration is to evaluate whether the system, as it has currently been implemented, provides correct results concerning two main issues. They can be summarized in: (i) identification of the different green and urban areas; (ii) their quantitative evaluation (length, width and surface).

In other words, inserting portions of urban and green areas whose size and location we have previously established within a starting image lets us know whether the results obtained at the end of the processing are consistent with the initial ones. The same result would not have been obtained if we had used satellite images of the same area relating to different periods to estimate the variations that occurred in a specific time interval. In this case, the processing would have been aimed at recognizing real variations, which, unlike the simulated ones, cannot be compared with the starting ones.

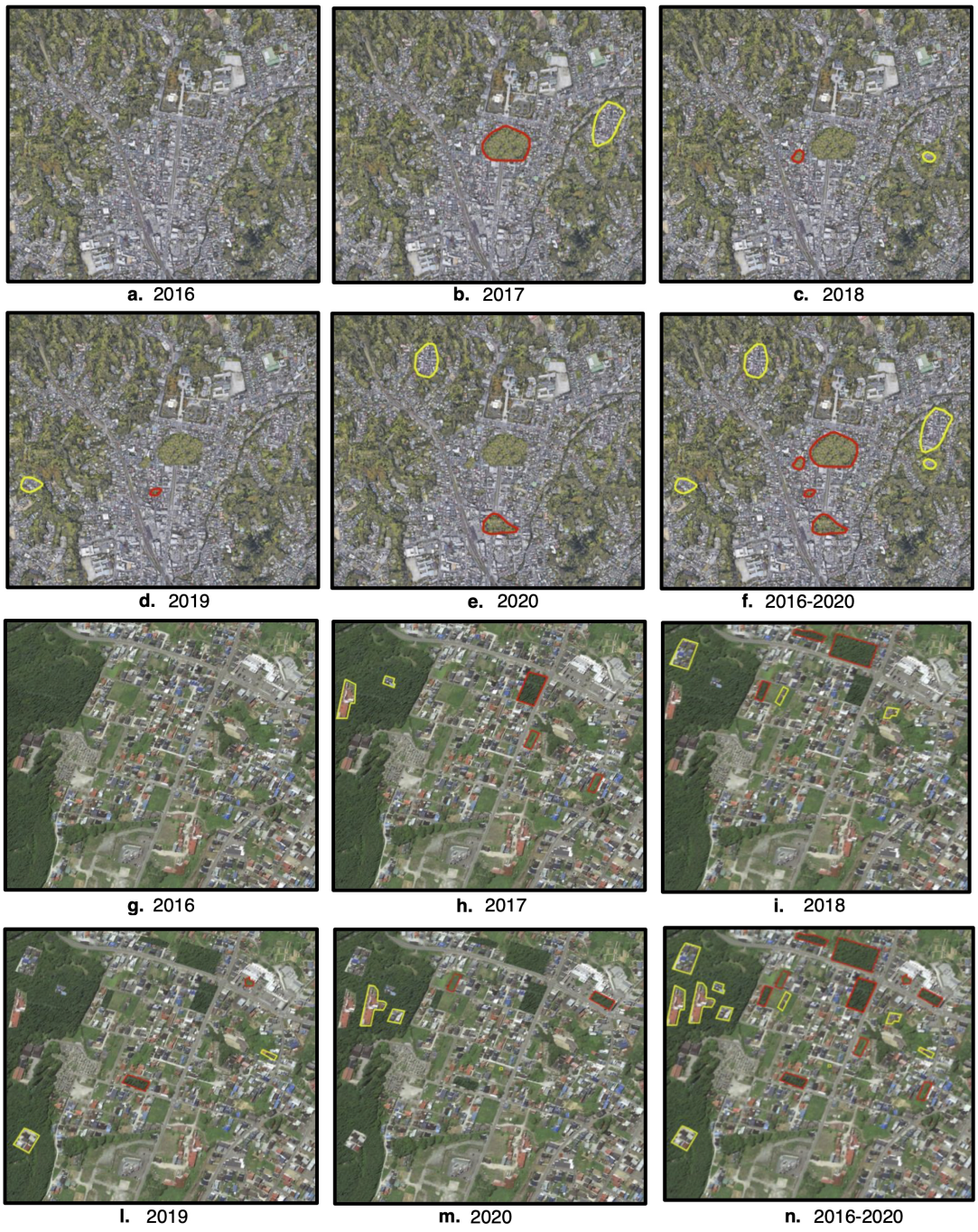


Figure 4.1: Some simulations referring to a hypothetical four-year period from 2016 to 2020. Green and urban additions in Kamakura and Ōhata cities are delimited in red and yellow.

4.4.1 Methods

The method simulates processing a set of images relating to the same area and different periods. We considered a five-element compute swarm at 1213x1069 pixel resolution for both the cities of Ōhata and Kamakura. By virtually introducing increases of urban and green regions in each image belonging to the swarm, we want to verify the application's capability to highlight them in a qualitative and quantitative way. The processing that allows the evaluation of the differences between green and urban areas is performed on the maps obtained at the end of the surface labelling process. First, a series of transformations of the original images were carried out to make them suitable for the set purposes. Each image belonging to the swarm has been named with the date of the acquisition in the following format: years-month-day.png.

Let us now consider the application of the method in Kamakura city. The first element of the swarm, named 2016-07-15.png, corresponds to the initial image, relating to the time t_0, to which no modification has been applied. In the following, from 2017-07-10.png to 2020-07-22.png, portions of green and urban areas have been introduced. The graphic editing made it possible to copy-paste these two types of surfaces into the same map.

Figure 4.1 shows the process applied to the swarm: the first element (image a) corresponds to the initial image at time t_0, in which no graphic modification has been used. In the following images, we can see the parts delimited in red (green areas) and yellow (urban areas) fictitiously inserted within them. The former corresponds to an increase in green spaces in the urban centre to improve livability conditions. On the contrary, the addition of urban areas to the detriment of green ones denotes a worsening of the healthiness of the urban centre. To verify the sensitivity of the software to identify the variations of surfaces of different sizes, from very small (a few pixels) to very large (hundreds-thousands of pixels) areas have been inserted. Similar considerations should be applied to the method used for the city of Ōhata, in which the preliminary phase of inserting the portions of

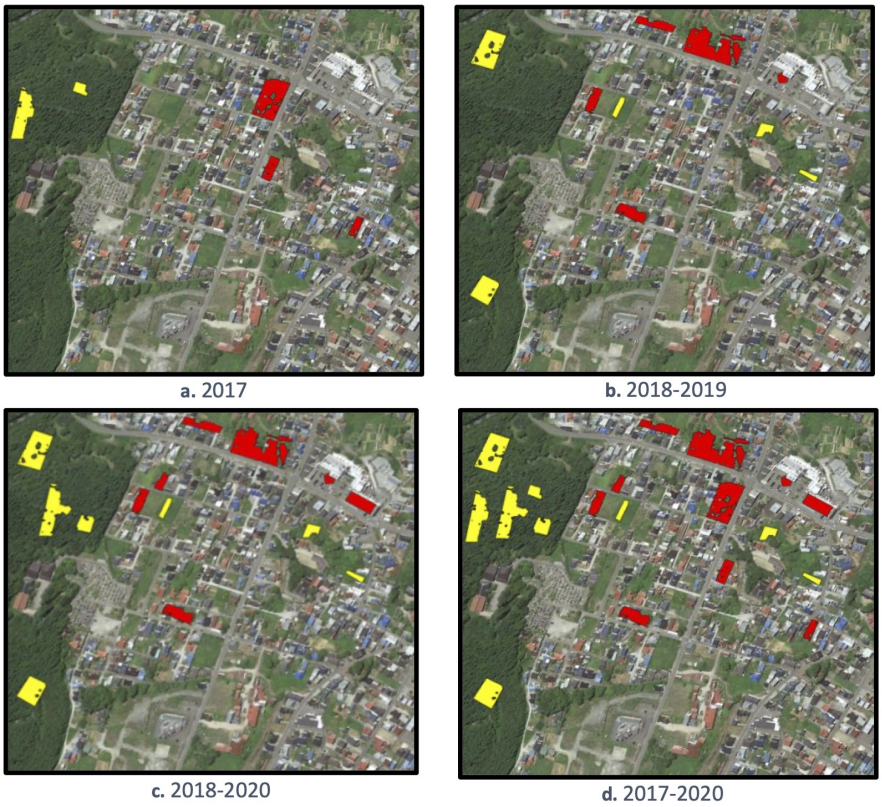


Figure 4.2: Results of the queries applied to the gradients of green (in red) and urban (in yellow) areas performed in the Ōhata city.

green and urban areas are illustrated in the images in Figure 4.1, from g to n.

The final results of the swarm processing were stored in specific fields of a JSON database for the subsequent phases concerning the execution of queries and their formatting and visualization in graphs and reports.

4.4.2 Results

Figures 4.3 and 4.2 show the results obtained at the end of the two simulations performed for Kamakura and Ōhata cities. Compared to Figure 4.1, all areas included in each image of the temporal range 2016-2020 were correctly identified. Notice that in most of the urban and green areas added to the image, extensive fragmentations are evident and, in some cases, holes that interrupt their continuity. Exceptions are some urban (yellow) and vegetal (red) areas in the four images of the 4.2 figure. This effect is not a software bug but highlights its correct execution

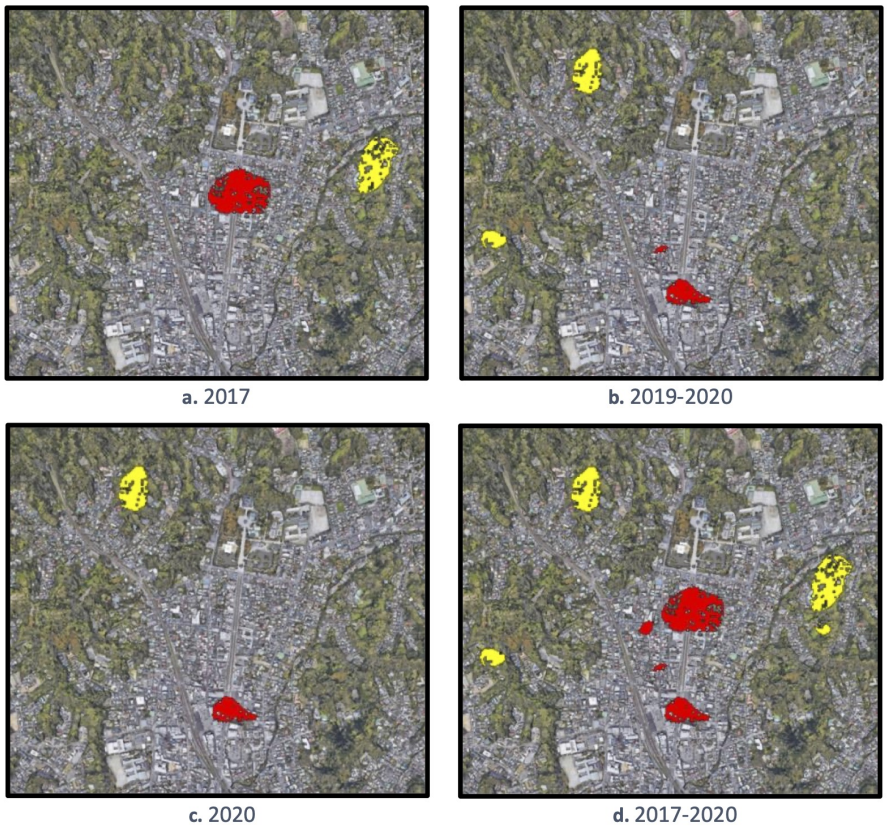


Figure 4.3: Results of the queries applied to the gradients of green (in red) and urban (in yellow) areas performed in the Kamakura city.

in highlighting the spatio-temporal variations.

There is a simple reason to explain the gaps in green and urban surfaces. The fragmentations in a yellow area are due to the portions of vegetation inside the urban appendage overlapping the image. The superimposition of a green area on a similar surface does not give any variation and not is, therefore, indicated by the yellow colouring of the respective pixels. The same goes for the green appendages that cover the pre-existing urban areas. In this case, the holes interrupt the red surfaces (variation from the urban area to the green area), denoting the presence of urban portions in the green surfaces pasted in the main image.

For both cities, the previously inserted green and urban areas have been correctly identified, and their tracing precisely reflects the initial position shown in the original figure (4.1). Furthermore, the *calcArea* function has made it possible

to calculate the surfaces of the new boundaries correlated to the variations that have occurred in one or more years. Their value, net of fragmentations due to the previously mentioned reasons, corresponds perfectly to what they originally had.

The operating procedure used to obtain each map with the relative variations and one or more years was divided into several steps. First, the calculation of the variations of the green and urban surfaces between the initial image (year 2016) and the following one (year 2017) was carried out. Subsequently, the same procedure was iterated until the last image of the swarm was reached, according to the following relationship: $year_{j-1} - year_j$. During the swarm processing, the surface gradients and the corresponding areas' length and width have been inserted in a JSON file. Based on these data and through queries, it was possible to create specific maps based on the variations relating to one or more years. By merging the data relating to one or more observation periods, it is possible to observe the gradients relating to a single year (Figure 4.3a-c and 4.2a), two years (Figure 4.3b and 4.2b), three years (Figure 4.2c) or the entire four-year period covered by the swarm (Figure 4.3d and 4.2d).

The algorithm in Figure 4.4 allows us to do what was described above and has been applied to the Ohata city dataset to obtain the results shown in Figure 4.2. The years on which to carry out the processing are established in advance by the user through the lists *listYears_index* with 0 $\leq$index$<$ 5 in which it is possible to insert single years or combinations between them. On the other hand, the *filename* list allows us to identify which period the saved images refer to at the end of processing. The *drawGradientData* function then allows us to create the maps based on the combinations of years previously established. Finally, the *drawArea* and *drawBoundaries* functions will trace the areas and boundaries of the surface gradients found in the time interval considered. The possibility of combining the data obtained from different observation periods has also made it possible to carry out a more accurate analysis through graphs and reports, illustrated in the next section.

```
ENABLED = True
DISABLED = False

def runQueries(gCoord,col,input_path,output_path,w,h):
    listYears_1=["2017"]
    listYears_2=["2018","2019"]
    listYears_3=["2018","2019","2020"]
    listYears_4=["2017","2018","2019","2020"]
    filename=["2017", "2018-2019", "2018-2020", "2017-2020"]
    drawGradientData(listYears_1,filename[0],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_2,filename[1],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_3,filename[2],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_4,filename[3],gCoord,col,w,h,output_path,ENABLED)

def drawGradientData(listYears,fname,gCoord,col,w,h,out_path,flag):
    '''Colour all urban and green areas with different colours.'''
    if flag == DISABLED:
        return
    img=Image.open(f"{out_path}/background.png")
    fillColor=col["White"]
    for k in range(len(gCoord)):
        year=gCoord[f"Boundary_{k}"]["Date"]["year"]
        if year in listYears:
            fillColor=selectArea(gCoord,col,k)
            drawAreas(gCoord,img,col,fillColor,k)
            drawBoundaries(gCoord,img,col,fillColor,k)
    img.show()
    JsonHelper.saveImage(img,out_path,f"gradient_{fname}.png",'png')

```

Figure 4.4: A snapshot of the queries getting after the swarm processing and applied to the new areas.

4.5 Real Case: analysis of thermal fields in the volcanic area of Etna

The method for the spatio-temporal analysis of natural phenomena relating to one or more portions of the territory has been applied to the Etna area, located in the southeastern sector of Sicily in Italy. The Etna volcano, formed about 700,000 years ago, has alternated explosive and eruptive activity and is still active. The objective is to verify, through the analysis of a swarm of images, the behaviour of heat zones, which significantly increase before and during volcanic events.

Knowing the areas of the volcano subjected to more significant heating, especially during paroxysmal events, can give valuable information to understand the internal dynamics of Etna. Areas with higher thermal values may indicate the rise of magmatic bodies or the mobilization of considerable amounts of magmatic

Figure 4.5: Paroxysmal eruptive activity of Etna in December 2015, taken from the West. Photo by Veronica Testa.

gases. A fully automated approach has replaced the repetitive work of processing each image with GIS software. We refer in particular to the following activities: (i) entering the values present in the metadata files; (ii) calculation of the parameters required to obtain the soil temperature; (ii) cropping of the part of the image to be processed. The developed application only requires users to download satellite images and metadata files.

The processing was applied to a set of images relating to the period from November 2015 to May 2016. The choice is related to the sequence of paroxysmal events with eruptive columns of the order of kilometers, which occurred in the first week of December 2015 (Figure 4.5), as described in the next section.

4.5.1 The Etna eruption of December 2, 2015

The paroxysmal event of 2 December 2015 is among the most violent of Etna in the last twenty years. After a progressive intensification in the evening, the eruptive activity of Etna culminated in the early hours of 3 December in a short but very violent paroxysm, with high lava fountains and an eruptive column several kilometers high. The excellent weather conditions made it possible to observe the event.

The peak of the paroxysm occurred between approximately 02:20 and 03:10 UTC when a sustained lava fountain reached heights of well over 1 km. The cloud of pyroclastic material was moved northeast by the wind, causing ash fallout in towns such as Linguaglossa, Francavilla di Sicilia, Milazzo, Messina and Reggio Calabria. At dawn, the eruptive activity had substantially ceased, even if some weak ash emissions still occurred [116].

The previous description of the event of 3 December 2015 represents a summary of the press release issued by National Institute of Geophysics and Volcanology (INGV) on the morning of 3 December at 9:00.

Three more paroxysmal activities occurred in the following days up to December 9. From 10 to December 31, further Strombolian activity of modest magnitude mainly concerned the New Southeast Crater (NSEC). Since January 2016, the eruptive activity of Etna has remained at very low levels. The period of stasis, which lasted several months, was interrupted by a new eruptive sequence, from 17 to May 26 2016, with the opening of two explosive vents at the base of the Northeast crater (NEC). On May 21, from 3:40 to 8:00, a third paroxysmal episode occurred, and a modest lava flow came out of a fracture on the southeastern side of the crater. In the following days, a series of Strombolian activities alternated and ended on the night of May 26. Subsequently, the volcano showed low activity in mid-July, when weak ash emissions were observed from the eruptive vent that opened in November 2015 [116].

4.5.2 Methods

Landsat 8 and Landsat 9 product data acquired by both the Operational Land Imager (OLI) and Thermal Infrared Sensor (TIRS) are delivered in 16-bit unsigned integer format [117]. The imagery's spatial resolution equals 30 meters, allowing an optimal representation of the variations of the heat fields relating to the entire volcanic building, whose surface is approximately 1,200 km^2. During the swarm image processing, the DNs of each pixel was converted from the original values, representing the amount of energy detected by the satellite, into Land Surface Temperature (T_B) in degrees Celsius. This procedure was carried out in three steps.

In the first phase, the Landsat Level-1 data has been converted to the Top Of Atmosphere (TOA) spectral radiance using the radiance rescaling factors in the MTL file. The formula to convert each DN (indicated with Qcal) into TOA spectral radiance (L_λ) is as follows:

$$L_\lambda = MLQcal + AL \qquad (1)$$

where ML and AL are the multiplicative and additive factors the MTL metadata file provides. The second step concerns the conversion of the Thermal band data from spectral radiance (L_λ) to the top of atmosphere brightness temperature in Celsius degree (T_{TOA}):

$$T_{TOA} = \frac{K2}{\ln\left(\frac{K1}{L_\lambda} + 1\right)} - 273.15 \qquad (2)$$

where $K1$ and $K2$ are the thermal constants retrieved in the MTL file.

The last step concerns the passage from the brightness temperature (T_{TOA}) to the Land Surface Temperature (T_B) with the following relationship [118]:

$$T_B = \frac{T_{TOA}}{\left[1 + \left(\lambda * \frac{T_{TOA}}{c_2}\right) * \ln(e)\right]} \qquad (3)$$

where:

$\lambda = wavelength\, of\, emitted\, radiance$

$c_2 = h * c/s = 14,388\ \mu m\ K$

$h = 6.626 * 10^{-34} J\ s\ (Planck's\ constant)$

$s = 1.38 * 10^{-23} J/K\ (Boltzmann\ constant)$

$c = 2.998 * 10^8\ m/s\ (velocity\ of\ light)$

$e = \left(\frac{NDVI - NDVI_{min}}{NDVI_{max} - NDVI_{min}}\right)^2$

To take account of the local temperature existing at the time of acquisition of the satellite image, for each sampling date, the value of the average temperature (T_{mLocal}), in degrees Celsius, measured in the locality was subtracted from that read from the sensor (T_{TOA}). Generally, the air temperature can be considered with a negligible error, equal to that from the ground [119]. In this way, the value obtained from the difference $T_{TOA} - T_{mLocal}$ allows us to obtain the temperature increase per day of the soil cleaned by solar heating. The air temperature data were obtained from the historical archive of the 3Bmeteo meteorological portal [120].

The values obtained with the previous formula, suitably grouped into classes at intervals of 10°C, make it possible to classify areas characterized by homogeneous temperature intervals. The values of the different heat cells have been reported in a table for each satellite image. It shows the surface area covered, in square kilometers, by each type of thermal cell. This elaboration, therefore, makes it possible to evaluate the thermal field induced by the underlying gases relative to a specific date.

4.5.3 Results

The images in Figure 4.6 show the results obtained from the processing of the heat fields relating to the activities of December 2015 and the end of May 2016. The first image on the top left concerns the situation of the volcano in 2015-11-17, about two weeks before the paroxysmal event of 2015-12-02. Intense deep heat fluxes concentrated mainly on the southeastern slope of the volcano can be seen,

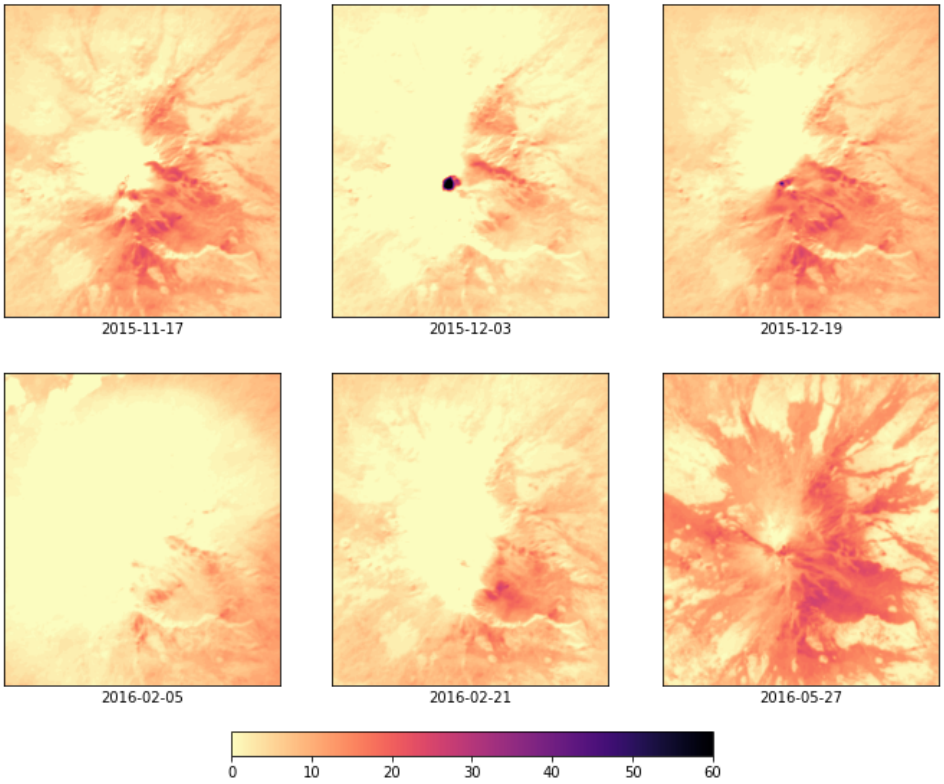


Figure 4.6: Heatmaps of the Etna volcano in the November 2015 - May 2016 period.

interrupting the snow cover located in yellow in the central part of the image. On the contrary, the following image, acquired on 2015-12-03 (GTM time: 09:36:10), shows reduced heat areas with lower gradients than the previous ones. The following image acquired on 2015-12-19 shows a rise in energy levels, according to the continuation of explosive activity from 10 to December 30 2015, as reported in the previous section.

The two images at the bottom left (2016-02-05) and in the centre (2016-02-21) show limited heat fields of minor magnitude. In both situations, the continuity of the snow cover, coloured yellow as it is at a temperature below 0°C, is interrupted by sporadic cells of endogenous heat. Also, in this case, the consistency of the heat fields is in agreement with the reports provided by INGV [116], which indicate a standstill from January to May 17 2016.

Finally, the last image on the right shows high heat levels on both sides of the

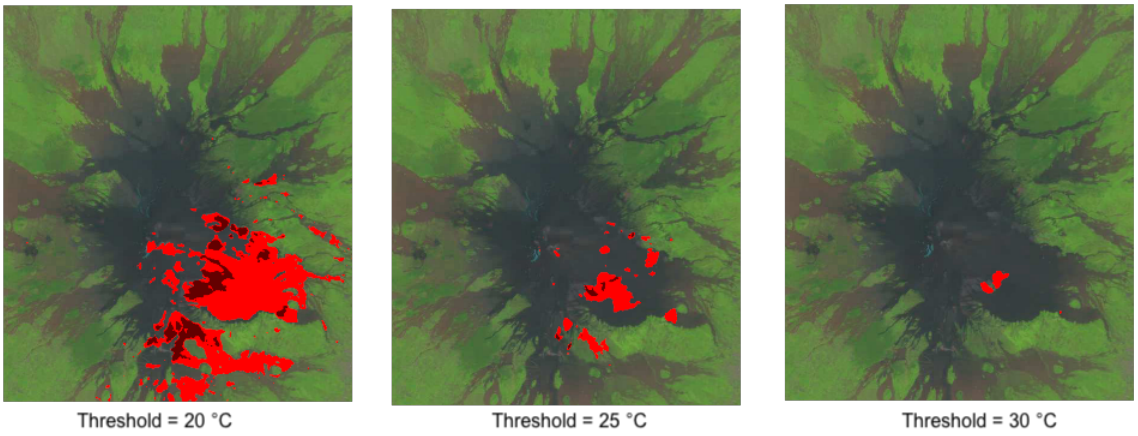


Figure 4.7: Frequency map of heat anomalies from November 2015 to May 2016 and for different threshold temperature values. The areas coloured in light red correspond to a frequency of five occurrences, while the ones in dark red to six.

volcano, with peaks corresponding to the entire eastern side of the volcano. The date and time of image acquisition (2016-05-27, GTM time: 09:35:40) indicate a snapshot of the volcano taken a few hours after the end of the eruptive sequence, which occurred from 17 to May 26 2016 (Figure 4.6).

One way to check which areas warmed up more frequently during the analyzed period is shown in Figure 4.7. To obtain the three frequency maps respectively for temperatures higher than 20 °C, 25 °C and 30 °C from the six processed images, the pixels that had temperatures higher than those indicated above and absolute frequency values of 5 (in light red) and 6 (dark red) has been filtered. The three images show that the surfaces of the thermal anomalies tend to decrease as the threshold values move to higher levels.

The images show that the areas most subject to thermal anomalies are those affected by tectonic structures (faults, extensional fractures, and eruptive fractures). Furthermore, the areas subject to heating are concentrated on the eastern flank, while on the other slopes, there is no trace of thermal anomalies in this specific case. This is in accordance with the well-known instability of the eastern slope sliding towards the opposite marine areas, dissected by the tectonic structures that spread from the Ionian to the continental areas. Using a more extensive dataset,

acquisition date	TEMPERATURE CLASSES (°C)				
	10°- 20°	20°- 30°	30°- 40°	40°- 50°	>50°
2015-11-17	259.79	63.50	4.23		
2015-12-03	190.51	7.34	0.37	0.10	0.38
2015-12-19	102.85	9.33	0.05	0.02	0.01
2016-02-05	22.11				
2016-02-21	254.62	30.86	1.03		
2016-05-27	0.20	163.06	156.79	24.55	

Table 4.1: Table of the heat areas expressed in square kilometers for each temperature class and image. The temperature values are reported in degrees Celsius.

this innovative approach can be a useful tool for identifying the potentially most dangerous areas in which deep magmatic gases are channelling towards the surface.

The table 4.1 shows the values, in square kilometers, of the heat areas relating to the medium-high temperature classes. Overall there is an increase in heat anomalies in the period prior to the paroxysmal activity (2015-11-17 acquisition date) or after the event itself (2016-05-27 acquisition date) except in the case of the paroxysm of 2015-12-02. In all other cases, there is a decrease in the heat cell surfaces. For the interpretation of the variability of the heat fields shown in the 4.1 table, see the Conclusions section.

4.6 Formatting and displaying JSON data

At the end of the image swarm processing, the acquired data were structured in textual and graphic form for each new green or urban area. The textual documents (reports) contain only the data necessary to define the phenomenon, leaving out some superfluous information. An example is given by the boundary and area coordinates relating to each theme which, for this type of representation, are not necessary. Both datasets, essential for the realization of the previous 4.3-4.2 queries, are not considered of any use in this phase, which has the purpose of im-

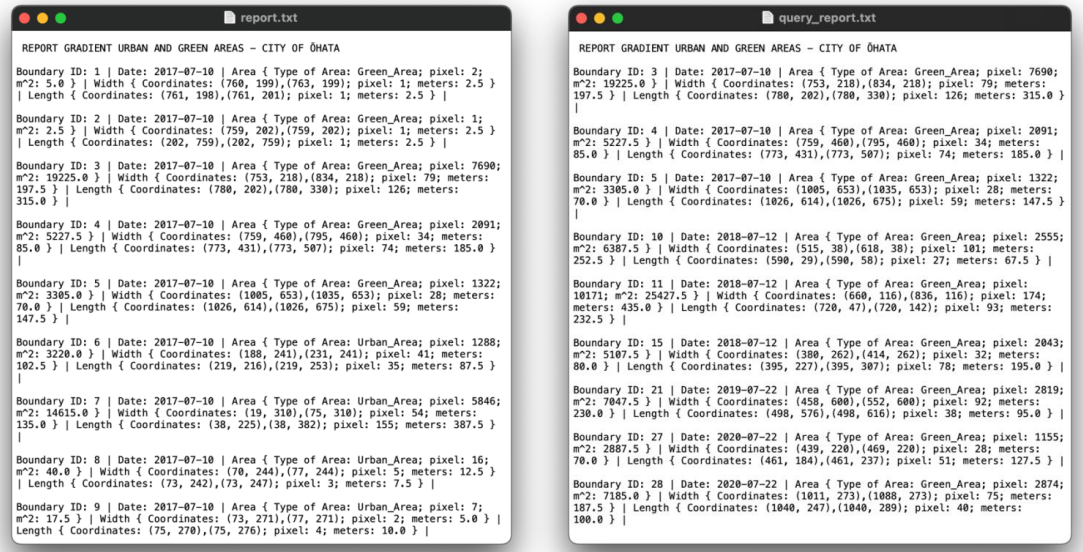


Figure 4.8: Two types of reports are obtained at the end of processing: (a) a complete report which contains the data of all the new boundaries identified in the image; (b) report showing data filtered by a query based on certain conditions (green areas with surfaces greater than 1000 pixels).

plementing graphics and textual queries. Figure 4.8a shows the complete report, which contains the results of the changes that occurred from 2016 to 2020. The new boundaries obtained in this time range are thirty four in total, even if the figure, for obvious reasons of space, shows only the first nine. The data formatting according to the predefined fields was done through the functions createReports, fillFields and getParameters imported from the graphs_and_reports.py module. The default report structure contains fields showing the corresponding values obtained during processing. They include: (i) the ID of the boundary; (ii) the acquisition date of each image belonging to the swarm; (iii) the type of area (green or urban) and its extension in pixels and square meters; (iv) the initial and final coordinates of the boundaries' width and length and their measurement expressed in pixels and meters.

In boundaries one, two and nine, the variations are minimal; in boundary two, they reach the value of an area pixel. These low values show that the application can recognize tiny variations (about 2.5 m^2) corresponding to private green areas

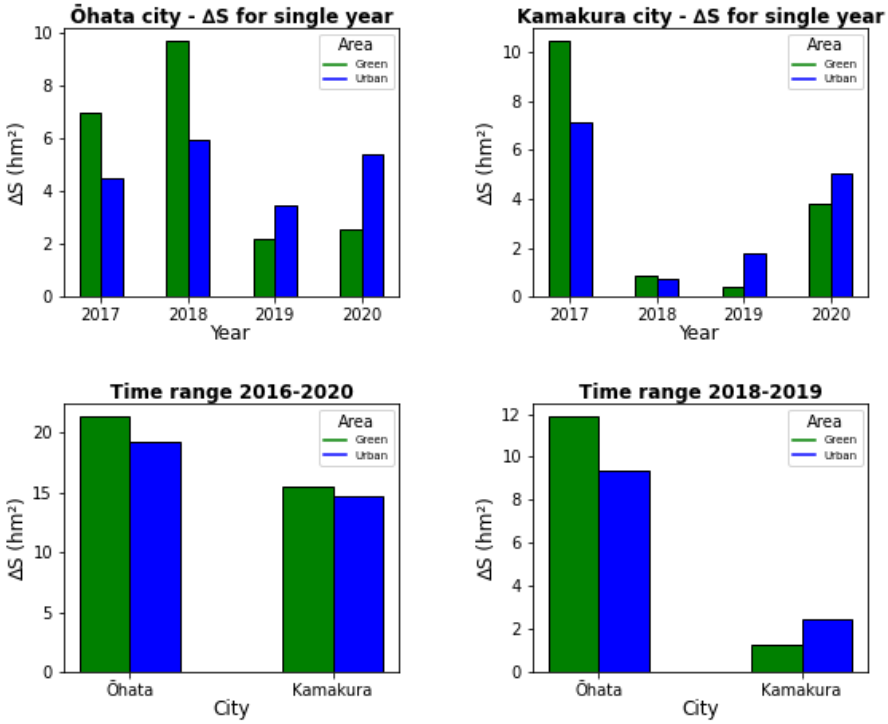


Figure 4.9: Bar chart of Ōhata and Kamakura cities. The images above show the variations, for each year, of the green and urban areas. In those below, a cross-comparison is made between the two cities in the intervals 2016-2020 and 2018-2019.

(gardens), individual buildings or urban infrastructure portions.

Figure 4.8b shows the result obtained at the end of the query, which selects all the green areas developed in the period 2016-2020, with a surface greater than 1000 pixels. In this case, the report is generated by the queryReport function in four phases: (i) access to the JSON database where the new green and urban areas are stored; (ii) selection of all green areas with a surface exceeding 1000 pixels; (iii) data structuring according to the predefined fields of the report through the createReports function assisted by the embedded routines; (iv) saving the report in txt format.

4.7 Data interpretation through graphs

Simultaneously with structuring the data in reports, the same was used to build graphs of various types. Based on the Matplotlib system library, a set of functions packaged in the graphs.py library have been implemented. In the specific case, bar graphs were used, which made it possible to interpret the evolution of green and urban areas in the two cities used for the simulated analysis.

The top images of Figure 4.9 show the increases in green and urban areas for each year starting from 2016-2017 to 2018-2019. In the images below, a cross-comparison was made between the Kamakura and Ōhata cities in the time interval 2016-2020 (left side) and 2018-2019 (right side). Although it is a simulation based on non-real data, the considerations that can be extrapolated from the graph analysis show interesting insights.

The surface increments (ΔS), expressed in square hectometres, referred to each year in 2016-2020, show different behaviour for the two cities considered. Both show a widespread development of the urban sprawl phenomenon in the years 2018-2019 and 2019-2020, in which there is a substantial increase in urban areas to the detriment of green areas.

A simple way to obtain the overall variation of green areas concerning urban ones (ΔA) is given by the difference $\Delta A = \Delta U$ - ΔG, where ΔU corresponds to the increase in urban spaces and ΔG in the green ones. Based on the previous formula, the urban areas' total increase (or green areas' total decrease) for Ōhata city equals 12,368.75 m^2 in 2019 and 28,800 m^2 in 2020. In Kamakura city, the overall increase in urban areas reached 13,650 m^2 in 2019 and 12,056.25 m^2 in the following year. In addition to comparing the variations of the green and urban areas of the same city, temporal correlations were carried out between different cities to monitor the phenomenon of urban sprawl. The two graphs at the bottom of the 4.9 figure show the results obtained. In the four-year time interval included in the 2016-2020 period in both cities, there was an overall increase in green areas equal to 20,875 m^2 (Ōhata) and 8,856.25 m^2 (Kamakura). Instead, in the years 2018-2019,

there was an opposite trend: although the city of Ohata had a marked vivacity in the construction field (93,612.5 m^2), the development interventions of the green areas compensated for the previous gap resulting even in excess (24,993.75 m^2) compared to the urban green existing prior to this period. The city of Kamakura, on the other hand, had the opposite behaviour: reduced increase in urban areas (25,118.75 m^2) but little attention to sustainable development (13,100 m^2), with urban areas increasing (12,018.75 m^2) compared to those prior to the two years 2018-2019.

4.8 Conclusions

Analyzing the spatio-temporal variations that a natural phenomenon produces on a given territory is a fundamental technique for environmental monitoring. This work proposes an alternative method integrable with previous ones, based on in situ sampling of physical parameters and analyzing satellite images using GIS [121]. The swarm processing of five satellite images aims to highlight, in qualitative and quantitative terms, the variations that occurred in a pre-established time interval.

The analyzed phenomenon concerns the evolution of urban sprawl in two Japanese cities, Ohata and Kamakura, which have different conformation of green spaces. The first has a predominantly two-dimensional development with agricultural land and urban meadows. In contrast, the green areas in Kamakura have a three-dimensional articulation made up above all wooded hills.

A simulation was carried out by inserting, in both cities, portions of green and urban areas in the four years following 2016. In this way, it was possible to verify the reliability of the results by knowing in advance the variations simulated from 2016 to 2020. The results show that the application cannot only ideally locate the previously added areas but also accurately calculates the values of the surfaces and their development in width and length.

Furthermore, unlike traditional GIS, the execution does not require operator intervention until the final step, in which the results are presented in graphs and

reports. The interpretation of the data in graphical and structured form makes it possible to deduce essential considerations on the evolution of urban sprawl relative to both unitary time frames and cumulative intervals (2016-2020 years). The ability to query data saved in JSON format allows us to obtain additional information compared to traditional GIS analyses performed on satellite images' temporal snapshots. Further potential related to this analysis is highlighting variations in different cities induced by local or global phenomena.

At the local level, negative variations in green spaces can be determined by natural phenomena (landslides and spontaneous fires) developed within the city. Even anthropic factors, such as building speculation, can significantly reduce urban green space over the years. Policies oriented towards sustainable development can instead generate positive increases in vegetation. Comparisons between the same city or contiguous urban areas over several years can confirm the matrix of these variations. Significant changes in the balance between green and urban areas in different cities worldwide can be related to global phenomena that affect large planet areas, such as pandemics and desertification. Also, in this case, the comparative analysis of image swarms can highlight the global nature of these variations.

The previous techniques have been applied to a real case to define, qualitatively and quantitatively, the variations of the heat fields of the Etna volcano during paroxysmal volcanic activities. The objective is to evaluate the potential and the applicability of the techniques used to different natural phenomena.

The results obtained at the end of the processing allow us to interpret the variation trend of the heat fields during the two paroxysmal phenomena of December 2 2015, and the eruptive sequence of 17-26 May 2016. Heatmaps of Figure 4.6 show variable heat fields before and after the two paroxysmal events.

The paroxysmal event of December 2 2015, one of the most violent of Etna in the last twenty years, was preceded by high energy levels, as shown by the image from 2015-11-17, where the intense heating is mainly localized on the eastern side of the volcano. Conversely, the following image acquired a few hours after the

paroxysmal event on the morning of December 3, shows a significant decrease in energy with significantly reduced heat areas compared to the previous ones. This phenomenon can be explained by a temporary depletion of energy confined to superficial levels due to the explosive paroxysm of the previous night.

According to the press releases by INGV [116], from 10 to December 31, other explosive activities occurred in correspondence with the New Southeast Crater (NSEC). Based on the previous sources, the 2015-12-19 image confirms that significant portions of the volcano are still subject to warming. In the same way, the following two images (2016-02-05 and 2016-02-21) confirm the period of stasis and slow recharge of the volcano, which since January 2016 has remained at low levels[116]. They show low energy levels with reduced heat cells set up on small portions of the volcano's eastern flank. The latest image acquired a few hours after the end of the eruptive sequence, from 17 to May 26 2016, shows very high energy levels, with heat fields extending to significant portions of the volcano. The previous observations are confirmed by the data provided by the table 4.1: the areas of heat show significant increases before and after the two paroxysmal events of December 2015 and May 2016, with a maximum found in the image sampled after the eruptive sequence of May 2016 (2016-05-27). In this case, in addition to the considerable amounts of volcano areas subject to thermal increases, there is a shift of the maximum surface values towards the higher temperature classes (20°-30°, 30°-40° and 40°-50°). Similarly, from January to May 2016, the heat fields remained at low levels, with a minimum found on 2016-02-05 with only 22.11 square kilometers of areas affected by heating belonging to class 10-20°C. In perspective, the proposed method is reliable for tracing the evolution of a wide range of natural phenomena and providing an interpretative key for analysing the mechanisms in progress.

Chapter 5

Refactoring techniques for the GIS application

5.1 Introduction

In recent years, Geographic Information Systems (GIS) had a growing diffusion and considerable technological development. The ease of use of their software interfaces allows different data such as roads, buildings, and vegetation to be shown on a map and related to each other (Figure 5.1). GIS software for studies involving geospatial analysis applied to bitmap images requires a preliminary tracing of the boundaries that delimit the different areas on the map [55, 56]. The operations are carried out with freehand tools (point, line, polygon), moving the analysis from the raster domain to the vector one [122]. This transposition is necessary since the different raster operators present in GIS systems provide good support for spatial analysis [123] but do not represent an alternative procedure to the vector one. After drawing the boundaries, the classic vector approach requires considerable work: joining the surfaces that overlap or share a common boundary in a single area is necessary. In the ArcGIS application, the Dissolve Boundary tool is responsible for performing this task to optimize the geometry of the boundaries. Then it proceeds with the surface labelling and the calculation of the main geometric parameters (length, width, and area), storing them in the table that contains the values of

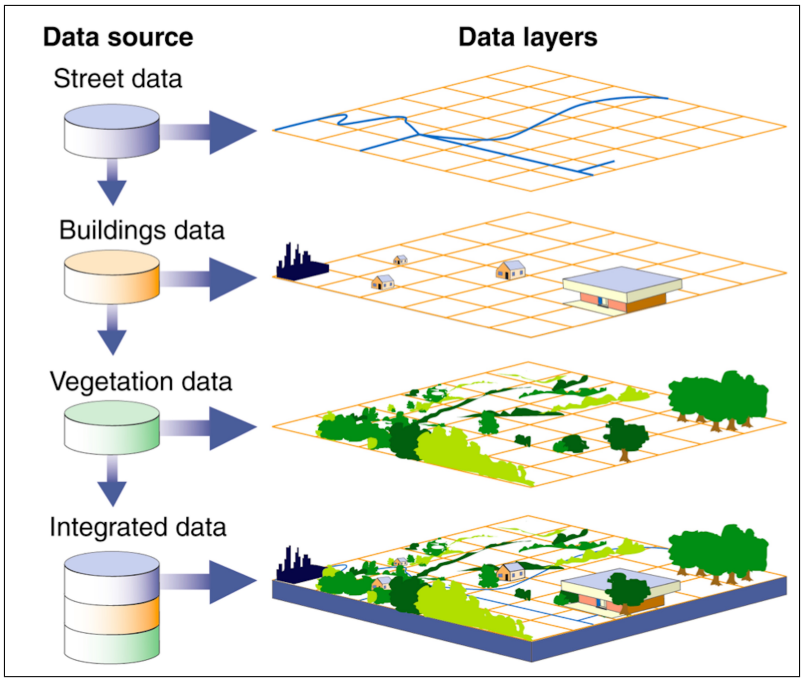


Figure 5.1: The concept of Geographic Information Systems (GIS). Illustration courtesy U.S. Government Accountability Office.

each vector shape. It is possible to add additional parameters obtained through the typical operations of spatial analysis to structured data: (i) use of buffers, i.e. areas that cover a given distance from a point, line or area feature, to perform a series of operations on the elements present inside them; (ii) interpolation of points relating to specific measurements to predict the values assumed in correspondence with new positions; (iii) application of set operations (union, intersection, difference) to highlight common characteristics between layers of the same type and not.

After these operations, there is a correspondence between the initial vector shapes and different datasets available for subsequent analysis based on SQL queries. They allow us to search and select subsets of features to extract helpful information from the image. The workflow just described raises a question: is it possible to perform spatial analysis directly in the raster domain without any vector transposition? Moreover, is it conceivable an automated procedure capable of carrying out a complete parameterization of the territory without any external intervention?

Recent solutions proposed to calculate geometric parameters (length, width, and area) of urban and green surfaces present within cities have confirmed both questions [124, 22]. The two types of areas were distinguished and delimited, starting from an input raster image. Next, through innovative algorithms, the geometric parameters of each green and urban body were calculated. Finally, on the obtained datasets, queries were performed in JSON format. This fully automated procedure is a helpful tool in the sustainable planning of a territory in which it is necessary to distinguish urban areas from green ones and calculate the different parameters.

In the proposed case study, the test was performed in two representative areas: Kamakura in Japan and Acireale in Italy [124]. The choice was due to the morphological characteristics of the two sites: the articulated and complex development of the green boundaries of Kamakura is the exact opposite of the simple and linear one of Acireale. In the first case, green spaces consist of wooded hills that develop in branched morphologies within the urban area. In the second case, however, they mainly consist of two-dimensional agricultural fields with regular geometric shapes. This structural contrast seems adequate to reveal the potential of the application in different conditions of complexity. The results confirmed the software's good adaptability to different environmental situations, avoiding the overfitting phenomena [124].

However, the current monolithic application has limitations in performing single or multiple images at increasing resolution. The temporal study of some phenomena may require processing applied to a swarm of them. We mainly refer to the geospatial analysis applied to urban agglomerations to evaluate the decrease in green spaces or similar phenomena that require a comparison between satellite imagery relating to different periods. The study of the desertification process is another example that necessitates the simultaneous processing of image swarms. The algorithms performed during processing require adequate computational resources, especially those that have to scan, pixel by pixel, the entire image or calculate the maximum and minimum values of x for the same y and vice versa

[22]. Therefore, improving performance to large input loads or single images of medium-high resolution is necessary.

The work aims to create a system capable of efficiently processing a swarm of images to monitor and interpret the evolution of particular natural phenomena. The method is based on comparing images of different periods to evaluate the changes that have occurred, in space and time, in a specific area. In this case, the method's novelty and significance are closely related to the challenge between the traditional monolithic system and the microservices one. It is not a trivial application of the microservices technique. Many architectural and implementation choices are made innovatively and strategically for optimal problem resolution. The external parallelization of containers and the exchange of data through an asynchronous messaging system requires their "ad hoc" use.

The transformation of the initial monolithic system into a microservices one makes possible some types of analysis in different fields (urban planning and geophysics, just to name a few) that were previously not possible for the time required. Performance analysis and evaluation of flexibility and portability can give valuable indications of the best compatibility between the two systems.

5.2 Related Work

Geospatial analysis is a set of methods and technologies that can extract information from geographic data. The application of these analysis concerns many fields and different purposes: (i) to understand the space-time dynamics of COVID-19, an essential factor for its mitigation, as it helps clarify the scope and impact of the pandemic and can aid community decision-making, planning and action [125] (ii) to analyze the equity of access to community goods and services through a GIS-based network analysis in combination with the statistical analysis of socio-economic data [126]; (iii) to support exploration and analysis within complex geospatial environmental data through the GCPC (Geo-Coordinated Parallel Coordinates) [127]; (iv) with a set of GIS indicators to evaluate accessibility to greenspaces for

sustainable planning in a dense urban context such as the city of Catania [128]; (v) aggregating the geographical areal data into hierarchical clusters based on the spatial similarity measured by distance to explore the global features and detailed characteristics [79]; (vi) applying geospatial technology in the field of geosciences to define the relationship between lithology and tectonic structures and create the lithological and structural maps for the metamorphic complex of Sinai [129].

The classic approach to the geospatial analysis of a given territory consists of a set of vector operations, usually carried out using GIS applications with a graphical interface and described as follow: (i) trace the boundaries between the different surfaces; (ii) calculation of the main geometric parameters of each theme (area, width, and length); (iii) SQL query execution [62, 63, 130, 131]. But this type of analysis, based on the use of freehand tools, creates a series of errors, like inherent or source error and operational or introduced error [132, 133] which produce different effects [134, 135] and propagate in the geospatial modelling phase [132, 136].

Another strategy to make the information contained within raster maps more accessible and usable is to extract the characteristics of interest and convert them into geospatial vector data [137]. These operations were performed by scanning maps in the past, but currently, the main GIS software has tools capable of converting from raster to vector and vice versa. However, these operations have some consequences: (i) vector-to-raster conversion is a process accompanied with errors [138, 139], classified into predicted errors before rasterization and actual errors after that [140]; (ii) for the raster-to-vector conversion, there are several commercial software but usually require human intervention except on simple cases [141]. The errors introduced are represented by the aliasing effects, the shift and superposition effects and the texture and text effects [142].

The freehand tracing of the geometric features (polygons, lines, points) and the raster-vector conversion involves introducing errors that reverberate in the subsequent spatial analysis. One way to avoid these errors and speed up the required tasks is to perform modelling and spatial analysis directly in the raster

domain, without any transposition or conversion in the vector field.

Some recent technologies, such as those based on microservices, allow it to move from traditional monolithic implementations to those based on independent components [143] which run each application process as a service. Switching between these two architectural styles brings benefits in terms of performance [144], scalability [145, 146], fault tolerance [146], and more isolation of the software components [68].

Also, in the field of GIS, various applications have followed this development trend. One field of application is risk analysis in the transport of hazardous materials, where a QGIS-based GIS microservice combines, in a map, information on the vehicle position with further data [147]. Many sectors apply similar architectures for several scopes: (i) the GIS application service of the electricity business where the microservices optimization combines the architecture advantages with the grid business. This approach allows overcoming the intrinsic limitations of the traditional single application system [88]; (ii) a specialized web-GIS based on a microservice architecture that provides analytical control of the disturbed component of geomagnetic field variations [148]; (iii) an IoT Smart City platform based on a microservices architecture that includes, among others, the Services that expose the District Information Models based on the data models of the Territorial Information Systems (GIS) [149].

5.3 Approach

The refactoring activity aims at preparing the application for a "migration" towards a microservices-oriented architecture. This activity made it possible to maximize the reuse of the parts of code in common between the various algorithms, providing greater modularity to the initialization (JSON file) and image loading phases.

The inclusion of helper modules allowed better management and isolation of components. The main components are: (i) JsonHelper for the management, ex-

traction and use of data stored in the JSON files provided as input; (ii) ColorHelper for color management and actual coloring of green-urban areas and boundaries; (iii) ImageHelper for loading and representing the output image based on the Pillow library; (iv) ProfilingHelper for profiling and measuring the performance of the application.

Another component introduced is the command module that acts as an orchestrator between the various phases that the application performs. The next phase of migrating to the microservices architecture used Docker to implement microservices and Apache Kafka to manage data streaming over a pipeline. We now explain the main concepts and functionalities of these technologies. Small independent services that communicate with each other through well-defined APIs constitute a microservices architecture. The benefits are the ability to scale and develop applications quickly and easily. Additional factors that make this approach preferable relate to flexibility, ease of deployment and resilience.

A monolithic architecture consists of a single indivisible block, based on three main components: a client-side user interface, a server-side application, and a database. Due to its rigidity, making changes becomes difficult and time-consuming for the entire system. Instead, in the microservices architecture, there is a system fragmentation in many isolated and independent units that carry out each process as a service. They allow scaling and developing applications faster and easier by adding new features and improving existing ones. There are two primary ways to enable data sharing between containers, based on synchronous and asynchronous techniques. The first method allows data sharing through programming interfaces (API). The asynchronous way involves data replication in an intermediate archive managed by an event streaming platform, like Apache Kafka. In this case, the services transmit their data to the system to subsequently share them with others.

5.4 Code Optimization Strategies

Processing satellite imagery can require a considerable amount of computational resources, especially considering image swarms needed to estimate the potential territory changes in a given time interval.

Therefore, we put forward a solution based on microservices in order to increase the throughput of the previous monolithic solution. The classic code restructuring solution was initially adopted to reduce computational complexity. During this phase, significant portions were rewritten with functional syntax to improve the readability of the code, producing a hybrid imperative-functional implementation.

The final microservices architecture supports parallelisation and uses a producer-consumer messaging system to exchange and synchronise data. We analyse the performance when considering the three main factors: input load, image resolution and architecture-related overhead, and show the scenarios where a balance between them can be found.

5.5 Refactoring from imperative to functional programming

Different algorithms were modified before porting to microservices to improve their performance and solve problems. The goal was to eliminate redundancies, streamlining some parts of the code through functional programming constructs. In particular, the algorithms for calculating geometric parameters (calcWidthAndLength and calcArea) have been rewritten. The hybrid implementation, with functional constructs within the imperative ones, was focused on exploiting potential based on iterators, closures and the MapReduce paradigm. The latter is mainly used in Big Data applications where processing is performed on large amounts of data and is suitable for processing based on multiple images.

The iterators allow iterating a sequence of data in lazy mode through the next

```
1   def calcWidth(bC,i,j,X_MinMax,maxV,minV,img,lstW,s):
2       widthValues=0
3       w=list(map(lambda x_1,x_2: (x_1[0],x_2[0],abs(x_1[0]-x_2[0]),x_1[1]), maxV,minV))
4       w_lst=list(map(lambda x_3: x_3[2],w)) #list that contain all y
5       max_W= max(w_lst)
6       m=next((z for z, c in enumerate(w) if c[2] == max_W), None)
7       '''(Xmin,Xmax,Width,y)'''
8       widthValues=w[m]
9       lstW.append((f"Boundary_{j}",max_W))
10      X_MinMax.append([f"Boundary_{j}",(minV[0],maxV[len(maxV)-1])])
11      JsonHelper.geometricParametersAsJSON(img,bC,i,j,widthValues)
12
13
14  def calcArea(bCoord,c1,img,col,endBoundary,lstX,px_area,lstArea,k):
15      x_1=c1[0]
16      if len(x_1)!=0:
17          lstX.append(len(x_1))
18      if endBoundary==True and len(lstX)!=0:
19          px_area=reduce(lambda a,b: a+b,lstX)
20          JsonHelper.setValuesArea(bCoord,lstArea,px_area,img,k)
21
```

Figure 5.2: Hybrid implementation of the calcWidth and calcArea functions with imperative and functional syntax.

function, which, in our case, helps determine the maximum values of the length and width of the boundaries. The map function has been used on various occasions: (i) to extract the coordinates (x, y) of a particular colour present in the image; (ii) to count or colour specific pixels. In other cases, the reduce instruction is associated with the map function to calculate the area subtended by each boundary. Closures were used to set queries: the ability to dynamically generate new functions allows us to format the report fields that summarize, quantitatively and qualitatively, the characteristics of urban and green areas.

The 5.2 figure shows two examples of the map and reduce functions application for calculating the width and area of the boundaries. Both use the lambda function, which allows us to declare simple functions more compactly in the body of the main program. The map function in line 3 takes as input the lists containing the maximum and minimum values of x for each y. The output is a list containing the quadruples (Xmin, Xmax, abs (Xmax-Xmin), y). In line 4, the map function extracts from the previous list its third value and inserts it in a new list. This value is equal to the length of the segments Xmax-Xmin, which will be used in the

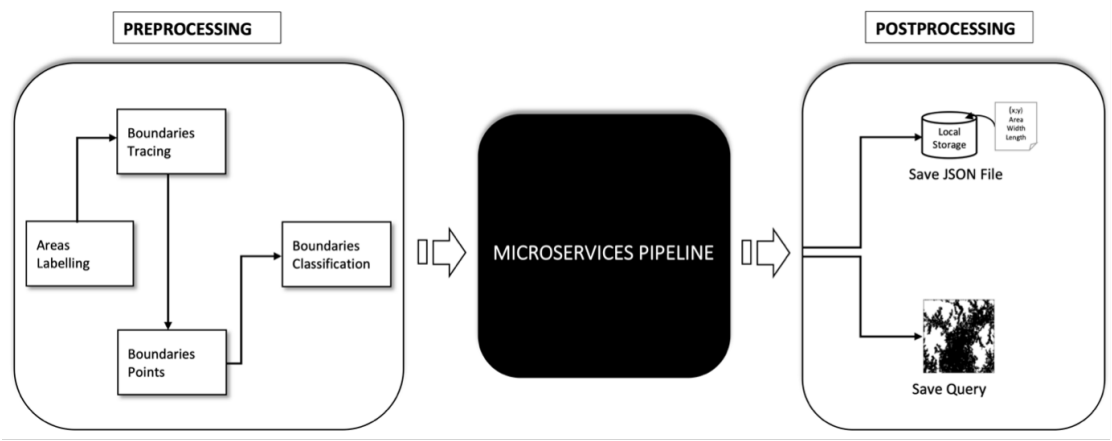


Figure 5.3: Black box diagram which describes the overall framework of refactoring to microservices

subsequent processing phases.

Finally, in line 19, the reduce function sum each element of the lstX list to its next. This process repeated for each horizontal row allows us to obtain the value of the area of the specific boundary.

5.6 Microservice Pipeline Framework

The application refactoring has been carried out using a pipeline of microservices. Figure 5.3 shows the context in which the "Microservices Pipeline" is inserted. The preprocessing phase concerns tasks such as labelling and the boundary recognition that separate green areas from urban ones, carried out without any refactoring. These operations are preparatory to acquiring the coordinates and their classification by ID. In the post-processing phase, illustrated on the right side of figure 5.3, the acquired numerical data are stored in a JSON file, and the queries performed during processing are saved in raster format.

5.6.1 Internal Framework

The following two diagrams describe the processing steps inside the black box. In Figure 5.4, each pipeline element corresponds with an algorithm phase. The

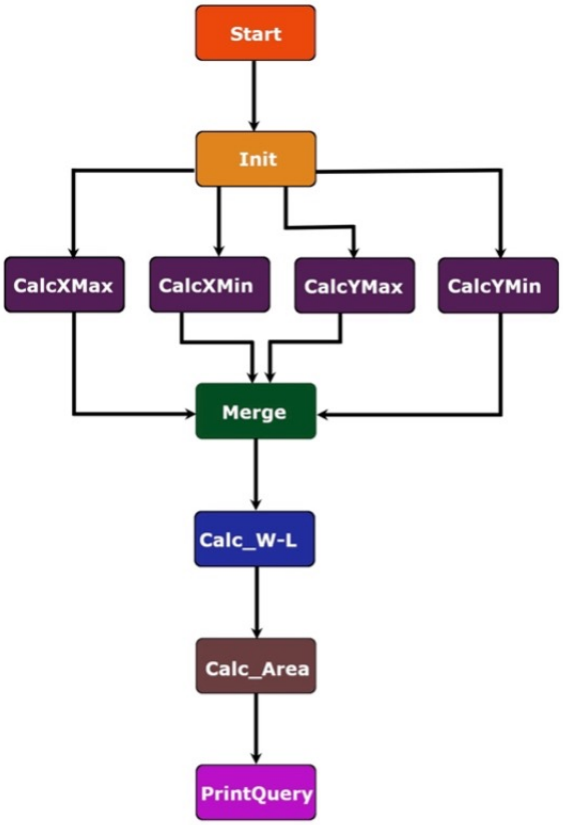


Figure 5.4: Microservices workflow

CalcXMax, CalcXMin, CalcYMax, and CalcYMin containers are independent and parallelised to obtain significant performance. The input data are given to each processing phase, sharing their output with the next merge step. The results are unified and made available for subsequent processing steps.

The choreographic approach adopted solves the problems of dependence and latency of the system that penalises the orchestration. This model uses a publish-subscribe system: producers publish messages, and consumers consume or pull that data.

Figure 5.5 illustrates the microservices pipeline and the interactions between them. The architecture of the microservices choreographic model is composed of the following components: Start, Init, CalcXMax, CalcXMin, CalcYMax, CalcYMin, Merge, Calc_WL, Calc_Area and PrintQuery. Processing begins with the Start microservice, which triggers the processing pipeline. The processing phase

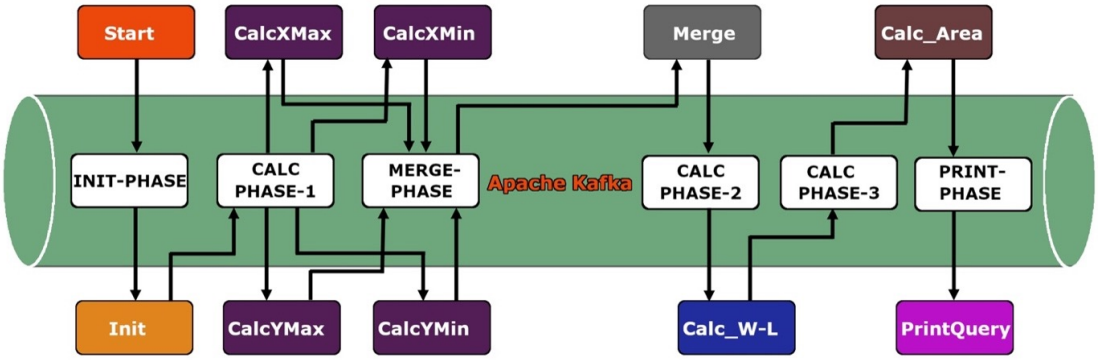


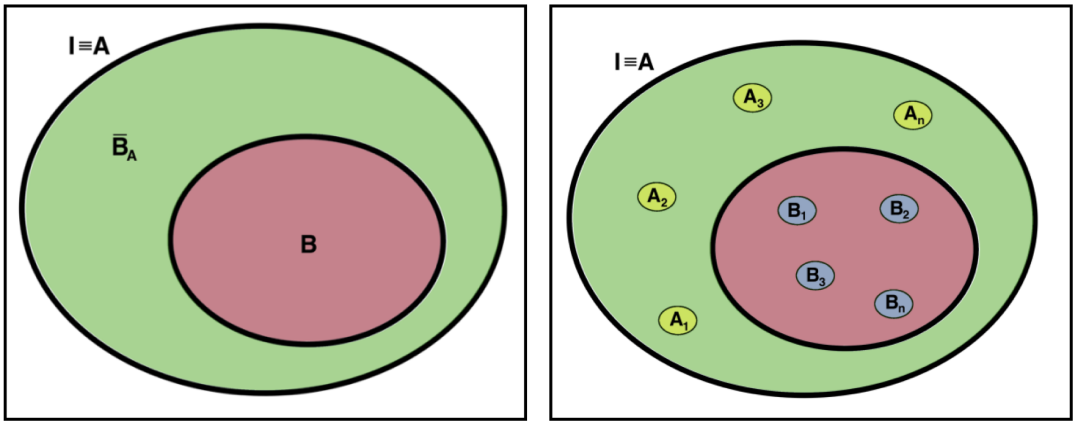
Figure 5.5: Microservices Pipeline.

will use the Data Transfer Object (DTO) generated by the previous service, which contains the paths to the preprocessing files on the shared volume. This first phase ends by sending the DTO via message on the Kafka queue to the INIT-PHASE topic.

The recurring reason for the subsequent processing stages concerns the transfer of the DTO to the specific Kafka queues that serve as a communication channel for the different microservices. Each of them pull and push the modified DTO in the respective queues, except for the CALC FASE-1 that performs in a parallel way the CalcXMax, CalcXMin, CalcYMax and CalcYMin microservices. In this case, processing differs from the previous ones for two reasons: (i) the four containers share the same input and output queues; (ii) a microservice which merges the output set received into a single file is necessary.

The last step of the pipeline concerns the queries processing, which generates the output images (Figure 3.6 c-d). This process concludes the cycle of activities carried out in serial-parallel mode by the processing pipeline and individual microservices.

The subsequent post-processing phase will save them with the numerical data in local memory.



a. Main subsets of the set of image points **b.** Further subdivisions into subsets

Figure 5.6: Representation of the coordinate sets related to the image. (a) The set image points (I) coincide with the one containing the points of the whole area (set A). Subset B contains all points relating to boundaries. $\bar{B}_A$ corresponds to the complementary set of A in B ($A - B$), i.e. only the points of the areas inside the boundaries. (b) Subdivision of sets B and $\bar{B}_A$ into proper subsets.

5.6.2 Computational Complexity

Computational complexity defines the resources needed for solving a problem characterized by a specific input in terms of time and memory. Evaluating the Big-O function for each algorithm is necessary to define the time complexity of the microservices pipeline. During these operations, the constant values of the execution time typical of primitive operations performed with limited machine language instructions have been neglected. We refer to arithmetic and logical operations, value assignments and comparisons, access to individual elements of a data structure and execution flow control operations (return, continue, break). Instructions of this kind require a constant time $O(1)$ which does not depend on the input size.

Analyzing the input that characterizes the main algorithms is a fundamental factor for understanding the procedures used to calculate time complexity.

Algorithm 1 *Calc_Area* algorithm

Input: *bCoord* #Dictionary of the boundary coordinates
Output: *nothing*

Function Calc_Area():
$pxArea \leftarrow 0$
$colorArea \leftarrow "Green"$
$n \leftarrow len(bCoord)$
for $k = 1$ to n **do**
    $m \leftarrow len(bCoord[k][X_{min}])$
    if $bCoord[k]["Type\ of\ Area"] == "GreenArea"$ **then**
        $colorArea \leftarrow "Green"$
    else
        $colorArea \leftarrow "Black"$
    end if
    for $j = 1$ to m **do**
        $p_{min} \leftarrow bCoord[k][X_{min}][j]$
        $p_{max} \leftarrow bCoord[k][X_{max}][j]$
        $pxArea \leftarrow calcPxArea()$
    end for
    $bCoord[k]["Area(pixel)"] \leftarrow pxArea$
    $pxArea \leftarrow 0$
end for

Algorithm 2 *calcPxArea* algorithm

Input: $colorArea, pxArea, p_{min}, p_{max}$
Output: $pxArea$

Function calcPxArea():
$x_1 \leftarrow p_{min}[0]$
$y_1 \leftarrow p_{min}[1]$
$x_2 \leftarrow p_{max}[0]$
for x_1 to x_2 **do** $\#\ t = x_2 - x_1$
    if $getpixel(x_1 + 1, y_1) == colorArea$ **then**
        $pxArea \leftarrow pxArea + 1$
    end if
end for
return $pxArea$

Figure 5.7: Pseudocode of the Calc_Area algorithm and the built-in calcPxArea function.

Analysis of the sets of boundaries and areas

At the end of the processing phase, the output file, in JSON format, will contain the n boundaries and the areas subtended by each of them relating to green and urban spaces.

Figure 5.6 a shows the sets of points, or image pixels, identified by a pair of coordinates (x, y). The set of the image points (I) coincides with the set of those belonging to the territory surface (A), so A is an improper subset of I, that is:
$A \subseteq I \wedge A \not\subset I$.
Conversely, the set of boundary points (B) is a proper subset of I (or A) expressed by the following relation:
$B \subset A \iff \exists(x, y) \in B \quad \wedge \quad \forall(x_1, y_1) \in B : (x_1, y_1) \in A \quad \wedge \quad \exists(\bar{x}, \bar{y}) \in A : (\bar{x}, \bar{y}) \notin B$

As shown in Figure 5.6 b, A and B derive from the unions of the respective subsets according to the following relations:
$\bar{B}_A \supset A_1 \cup A_2 \cup A_3 .... \cup A_n \wedge B \supset B_1 \cup B_2 \cup B_3 .... \cup B_n$.
Since all boundary points are also included in the occupied surface, the union between the area inside the boundary (A_1) and the boundary itself (B_1) will give the total area (A_{T1}) of the specific green or urban space. Therefore, for all n boundaries acquired during the processing phase, we will have the following:

$A_{T1} \supset A_1 \cup B_1$

$A_{T2} \supset A_2 \cup B_2$

$\vdots \qquad \vdots \qquad \vdots$

$A_{Tn} \supset A_n \cup B_n$

The set of points in the image (I), which coincides with the points of the entire territory surface (A), will be given by the union between the previous subsets:
$A \equiv I \supset A_{T1} \cup A_{T2} \cup A_{T3} \cup ....A_{Tn}$

calcPxArea Function:

$$3 + 1 + (t + 1) + t + t + 1 \Rightarrow 6 + 3t$$

Calc_Area Algorithm:

$$4 + (n + 1) + n + 2n + n(1 + (m + 1) + m + 2m + m(6 + 3t)) + 2n \Rightarrow$$

$$\Rightarrow 5 + 8n + 4nm + nm \sum_{i=1}^{t} 6 + 3i \Rightarrow 5 + 8n + 4nm + nm \left(6t + \frac{3t(t+1)}{2}\right) \Rightarrow$$

$$\Rightarrow 5 + 8n + 4nm + nm \left(6t + \frac{3t^2}{2} + \frac{3t}{2}\right) \Rightarrow 5 + 8n + 4nm + \frac{3}{2}nmt^2 + \frac{15}{2}nmt$$

Figure 5.8: Calculation of the time complexity for the *Calc_Area algorithm* and its built-in function *calcPxArea*, in basic number of steps.

Temporal Complexity Analysis

Table 5.1 shows the temporal complexity estimated for each algorithm. The greatest temporal complexity concerns the algorithm for calculating the area. Figure 6 shows the pseudocode of the Calc_Area algorithm and the built-in function calcPxArea. The latter is responsible for calculating the number of pixels for each horizontal line within the boundary.

Overall there are two nested loops inside the main one. The master of size n scans the boundaries within the list (Algorithm 1, row 5), while the second of variable size m (Algorithm 1, row 12), analyzes the data of the X_{min} list, with Length (X_{min}) equal to Length (X_{max}). The two pairs of values relating to X_{min} (x_1, y_1) and X_{max} (x_2, y_2) are sent to the function calcPxArea, which has another cycle of dimension t = x_2 - x_1 inside (Algorithm 2, row 5). Through this iteration, the function counts all the horizontal pixels between the values of the coordinates x_1 and x_2. The process is repeated for all horizontal lines that correspond to the surface bounded by the boundary.

To calculate the complexity, we must consider the three cycles just described. The first is repeated n times, equal to the number of green and urban boundaries identified on the map. In the second loop, there are m iterations, which corresponds

to the length of the list X_{max} or X_{min} since, for the same boundary, they both have the same number of elements. Finally, the number of comparisons carried out by the third cycle within the built-in function calcPxArea is equal, for each of the m rows and for the same value of y, to the t difference between the maximum value of x_2 and the minimum value x_1, i.e. t = x_2 - x_1.

Figure 5.8 shows the procedure adopted for calculating the complexity of the *Calc_Area algorithm* in number of basic steps. In the upper part, the time complexity of the *calcPxArea function* is obtained by considering: (i) three external assignments; (ii) one assignment, t+1 comparisons and t increments (for loop); (iii) t internal comparisons and a return statement. The temporal complexity for the built-in function is expressed by: $6 + 3t$.

The above value is used to calculate the *Calc_Area* complexity, which is called within the for loop for m times. The final result of the complexity in number of fundamental steps is expressed by:

$5 + 8n + 4nm + \frac{3}{2}nmt^2 + \frac{15}{2}nmt$.

The passage to the asymptotic complexity allows defining the upper limit of the number of iterations, expressed by the following relation: $O(n\ m\ t^2)$.

Similar behaviour to the above characterizes the printQuery microservice where the highlightsAreas function has the greatest time complexity, using the same technique described above to colour specific filtered areas based on size and type. As shown in Table 5.1, all the other algorithms have lower complexity. In particular, CalcXYMax-Min indicates the grouping of the four algorithms calcXMax, calcXMin, calcYMax and calcYMin. Each of them, with equal temporal complexity, consists of two for loops and a while loop. Their asymptotic complexity is given by: $O(n\ m\ t)$.

The $O(n)$ complexity of the algorithm relating to the Start microservice takes into account the size of the number n of images of the swarm processed by the pipeline.

Finally, the temporal complexity of the entire pipeline will be equals to the maximum complexity of the microservices included in it: $O(n\ m\ t^2)$.

Microservices	Temporal Complexity
Start	O(n)
Init	O(1)
CalcXYMax-Min	O(n m t)
Calc_W-L	O(n m)
Merge	O(n)
Calc_Area	$O(n\ m\ t^2)$
PrintQuery	$O(n\ m\ t^2)$
Full pipeline	$O(n\ m\ t^2)$

Table 5.1: Temporal complexity of the microservices pipeline

5.7 Experiment and Results

The refactoring concerns an application in which performed the spatial analysis in the raster domain. The aim is to create an automated system capable of distinguishing green and urban areas, calculating their main geometric parameters and querying the corresponding raster data model. A series of algorithms characterized by different computational complexity have carried out these tasks [22].

Figure 3.1 shows the various functions and their execution times. The calcXMaxOrMin and calcYMaxOrMin algorithms have the lowest performance at the resolution of 2502 x 2607 pixels. The refactoring and parallelization of these components aim to speed up the application and ensure some aspects such as isolation and scalability. The microservices implementation has been extended to the calcWidthAndLength and calcArea functions and to queries that have a functional dependency on the previous algorithms.

The realization of the pipeline profiling made it possible to evaluate the performance of refactoring to microservices, calculating the execution times for the entire

Algorithms	MONOLITHIC			MICROSERVICES		
	1213X1069	2416X2129	4255X3750	1213X1070	2416X2130	4255X3751
calcXYMax-Min	13.014	51.672	153.864	8.957	31.322	88.518
calcWL-Area	4.081	13.284	35.894	7.271	25.061	76.577
printQuery	0.64	1.341	2.168	1.857	6.228	15.889
whole application	18.702	68.798	197.115	24.666	78.616	214.384

Table 5.2: Execution times of the main algorithms for Monolithic and Microservice architecture at different resolutions. Performances are expressed in seconds and resolutions in pixels.

Number of images	Monolithic	Microservices	Deviations
1	18.784	23.201	-4.417
2	36.994	37.622	-0.628
3	55.317	46.632	8.685
4	74.764	59.532	15.232
5	94.486	72.158	22.328

Table 5.3: Performance for the monolithic and microservice implementation of different image streams at resolution 1213X1069 pixel. The last column corresponds to the difference in execution times between the two architecture. Performances are expressed in seconds.

application and each running microservice. Profiling the pipeline means calculating the difference between the current timestamp and the session_ID generated by the Init microservice.

The work was based on a series of tests carried out on different images relating to the same area of Kamakura. From a reference figure of 1213X1069 pixels, another four were extracted with the same resolution, translated by 200 pixels concerning the four cardinal points. The offset attributed to each image aimed to simulate a variable execution context to evaluate any deviations in the application of the algorithms. The results obtained were compared with each other and reported in tables and graphs.

Refactoring the monolithic version of the application also produced some performance advantages over the original implementation, as shown in Table 5.3. Each run on swarms and single images was performed five times to obtain averaged experimental data. Monolithic and microservices application performance tests were performed on a MacBook Pro (13-inch, 2018, Four Thunderlbolt 3 Ports) PC, with the following characteristics: (i) MacOS Big Sur Version 11.1; (ii) 2.3 GHz Quad-Core Intel Core i5 processor; (iii) Intel Iris Plus Graphics 655 1536 MB; (iv) RAM memory 8 GB 2133 MHz LPDDR3.

Table 5.2 shows the execution times for the different algorithms, while Figures 5.9 c-d-e show the variation trend at the different resolutions. Note that the calcXMax, calcXMin, calcYMax, calcYMin algorithms, globally referred to as calcXYMax-Min, show a more significant performance advantage for the container architecture than the classic monolithic one, raising as the resolution of the image increases. Conversely, the monolithic application performs better than microservices for the algorithms that calculate the length, width and area subtended by the boundaries, collectively defined as calcWL-Area. As can be seen from the graphs, minor variations occur in the algorithm that executes the queries (printQuery).

Graph of Figure 5.9 a shows the behaviour of the two types of architectures in a growing image stream. The monolithic approach, albeit limited, shows better performance than the microservices one, up to two input images. The reversal

trend occurs for additional input loads: the microservice implementation shows performance with more significant increases. Figure 5.9 b analyzes the differences between the execution times of the monolithic architecture and the microservices one. As shown from the numerical data shown in Table 5.3, this difference tends to diverge increasingly as the number of input images increases, reaching the value of about 22 seconds for a swarm of five images.

The algorithms had consistent behaviour for each element belonging to the swarm. In the previous query applied to the reference image (Figure 3.6 d), small green and urban areas were filtered. A similar effect was obtained in the other images without significant deviations from the initial behaviour.

Finally, an integration test allowed checking whether the changes introduced following the refactoring process to microservices have altered the correct functioning of the application. This type of test verifies the functional correctness in the interaction between multiple modules. Integration tests usually refer to an application workflow. It checks the correct behaviour of each object and the relationships with the other components of the application. The output produced by running the application before making any code changes is considered a reference image (source image). Verification compares the new output image produced with the source one after applying a pHash function to both. If the two results are the same, the test is declared passed; otherwise, failure. The integration test results confirmed the absence of any bugs or regressions introduced by refactoring to microservices.

5.8 Interpretation of Experimental Data

The estimate of execution times performed on the original monolithic version showed that the greatest execution times are related to the functions that calculate the maximum and minimum values of X and Y and to the algorithm that computes the area (Table 3.1). There is a functional dependence between them: calculating the surface, length and width of green and urban bodies presuppose the

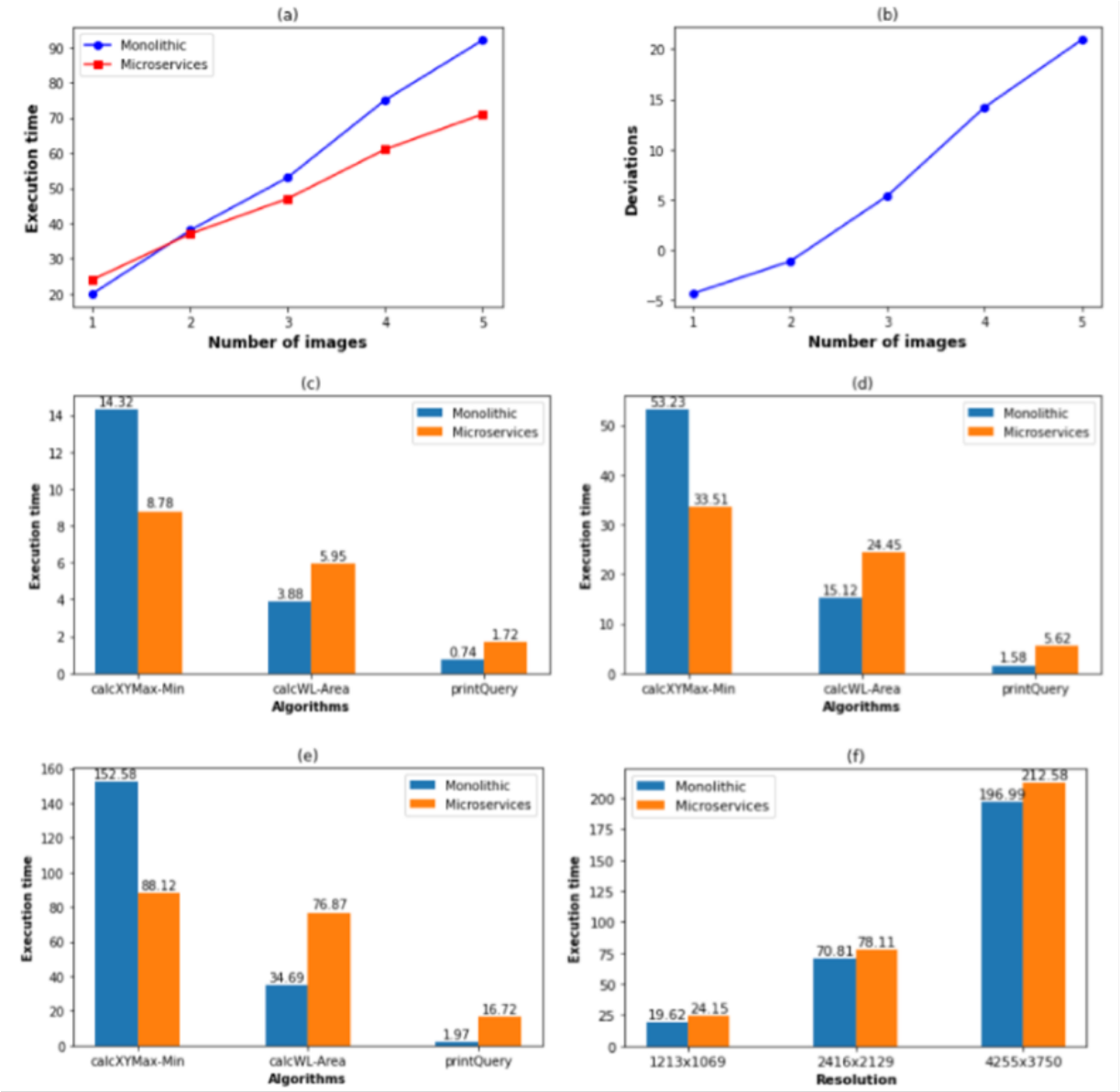


Figure 5.9: Graphs of monolithic and microservices architecture execution times of single and multiple images at different resolutions. (a) Comparison between monolithic and microservices architecture, at 1213X1069 pixels resolution, for increasing image inputs. (b) Trend of the difference between the two types of architectures related to the previous graph. Execution times of different algorithms implemented by monolithic and microservices architectures and different resolutions: (c) 1213x1069 pixels; (d) 2416x2129 pixels; (e) 4255x3750 pixels; (f) whole application.

availability of the values calculated by the calcXMaxOrMin and calcYMaxOrMin algorithms.

The strategy to make the application more efficient involves parallelising the calcXMaxOrMin and calcYMaxOrMin functions, whose code has been fragmented into the four corresponding algorithms. In this way, each of them is responsible for calculating a single set of maximum or minimum values of X/Y through the calcXMax, calcXMin, calcYMax, calcYMin functions, which are contained within the corresponding microservices.

The architecture manages an external parallelisation of the four containers with a further microservice (Merge) that coordinates the results and inserts them in the queue relating to the MERGE-PHASE topic. As shown in Table 3.1, the calcWidthAndLength function execution time is negligible compared to the algorithm that calculates the area. For this reason, it was decided not to carry out a further parallelisation of the two algorithms that deal with the parameterisation of the green and urban units: their execution times have in any case been grouped together in a single item called calcWL-Area.

Computational complexity was quantified in $O(n\ \ m\ \ t^2)$ for the entire pipeline. However, in the worst case, we must consider that the three variables have limited inputs in medium-high resolution images. In particular, the n and m variables depend on the number of boundaries identified within the territory. In other words, both are related to the greater or lesser articulation of green spaces within urban areas. As seen from Figure 5.6, the n variable corresponds to the boundaries set's cardinality ($|B|$). Furthermore, the m parameter is also correlated to the same set and is equal to half of the total cardinality ($\frac{|B|}{2}$) of set B. Each m boundary corresponds to the size of the Xmax or Xmin lists, that is, to half of the points of the same boundary. Finally, the quadratic term t has the width (w) of the whole image as its maximum value, whose order of magnitude is of the thousands of pixels.

The graphs and tables analysis highlights that the entire application's performance is conditioned by the resolution and volume of images processed. The

performance is lower or equal than the monolithic one, up to two inputs with resolution 1213X1069 pixels. Additional images involve a clear performance advantage for the containerized implementation, reaching a significant difference for a swarm of five inputs (Figure 5.9 a-b). Conversely, Table 5.3 shows that, as the image resolution increases, the monolithic architecture is more efficient than the microservices one, which reaches a gap of about 15 seconds for image processing at 4255X3750 pixel resolution.

The previous conflicting results lead to additional considerations. For a limited number of inputs (up to two images), the overhead associated with the microservices implementation makes the monolithic approach preferable to the latter. This is due to the own nature of architectures based on interactions between containers that introduce overhead related to a higher workload for operations, deployment, and monitoring. As services communicate, the resulting high number of remote calls can lead to higher network latency and processing costs. Furthermore, each microservices has to deal with several cross-cutting concerns like logging, metrics, health checks, externalized configuration, etc. [150].

5.9 Conclusions

In GIS applications, the results obtained with freehand tools are strongly affected by the operator's subjectivity and make it almost impossible to reach a high level of precision, as manually tracing outlines is difficult and time-consuming. Currently, a widely used procedure consists of firstly converting images from vector to raster, since the latter are more apt to get geospatial analysis in urban areas, and they allow adopting simplified procedures and reducing execution times.

We have presented the development of a geospatial analysis tool for raster images, doing without the vector techniques adopted by other main GIS software systems. The design of the tool was centered around a microservice architecture with special care to performance tuning. For each microservice, the main responsibility was chosen to ensure efficiency, isolation, and independence of the software

components. For all the microservices to work, the initial phase is about computing the main geometric parameters of the different areas in an image, i.e. the minimum and maximum values for each x and y of the boundary. In the preliminary monolithic implementation, the algorithms for computing such parameters have the greatest computational complexity.

A parallelization strategy was adopted to improve the performance, with the simultaneous execution of four functions (calcXMax, calcXMin, calcYMax and calcYMin) globally referred to as calcXYMax-Min. The results obtained show better performances of the containerized version, compared to the monolithic one, for processing more than two input images. The opposite behaviour has been observed for single or multiple inputs at higher resolutions, as the monolithic architecture shows the shortest execution times. The contrasting behaviour related to the input load and processed images resolution seems to be determined by the performance of the calcArea and printQuery algorithms, which negatively affect the entire tool. This is also confirmed by the estimated time complexity for the microservices pipeline, with the respective algorithms showing the worst results ($O(n\ m\ t^2)$).

The computational gain of the calcXYMax-Min functions and the respective containers is due to the external parallelization, which reduces their execution time. Accordingly, the microservices solution is advantageous when processing multiple lower-resolution input images despite the more overhead. Conversely, the sequential execution of the Calc_WL and Calc_Area containers is due to the negligible execution times of the calcWidthAndLength algorithm whose parallel containerized version would not provide any gains. Therefore, unlike the previous case, it is not possible to increase the workload to balance the overhead due to the microservice architecture. This has been observed for both single and multiple medium-high resolution images.

The analysis of the work leads to further conclusive considerations for possible future developments of the application, both in terms of additional features and potential performance improvements. Regarding the first point, the current

software recognizes two types of surfaces corresponding to green and urban areas. Further implementations could extend the scope to more themes and to different purposes. An example could be the classification and parameterization of the volcanic, sedimentary and metamorphic rocks in specific areas. Other possible applications could concern the analysis of space-time variations of some natural phenomena through creating monothematic maps divided into value classes. Examples of these techniques involve calculating the dimensions of the heat cells in volcanic areas and evaluating the vigour degree of the surfaces covered by vegetation in rural areas. These types of analyzes applied to image swarms relating to specific time intervals allow us to define the evolution of volcanic and desertification phenomena in progress.

Potential margins for performance improvement can also be obtained through the internal parallelization of some software components. The MapReduce paradigm and Machine Learning techniques could be helpful tools to ensure greater linearity and readability of the code and extend recognition and automated parameterization to several different themes.

Chapter 6

Conclusions

The path followed in the three years of PhD work had the environment and the related natural phenomena as a common denominator. Living in a healthy environment and minimizing the risks due to endogenous and exogenous phenomena are the main objectives of environmental monitoring.

The abundance of collected data characterised by different spatial and temporal resolutions and the ability to make such data easily available to many researchers leads to different and much more advanced analysis opportunities from past ones. These advantages are offered by satellite images which in recent years have proved to be a complementary tool to in situ analyzes to interpret natural events. However, the result's quality depends on the proposed solutions to describe the phenomenon correctly. An essential role is that of the techniques used, which, in addition to extracting critical information, must also operate efficiently.

The activities carried out were aimed at defining innovative methods, based on direct and indirect monitoring of environmental parameters, capable of contributing to the reduction of risks in the volcanic and seismic fields and controlling the phenomenon of urban sprawl in cities. An alternative model to the previous early warning systems for high-intensity seismic and volcanic phenomena has been proposed. Generally, their implementation is based on monitoring the main geophysical precursors represented by Volcanic Tremor, Seismic Swarms, Ground Temperature, Ground Deformation and Radon Concentration. Abnormal values

of one or more of them give a pre-alarm to the apparatus responsible for managing population risk events. In several cases, however, the increase in values has not been accompanied by the paroxysmal event.

Our approach was based on some simple considerations: the difficulty in predicting natural phenomena depends on their complexity due to the numerous variables involved. A technique that could prove effective in predicting paroxysmal natural events should consider the single contribution of each precursor and their interactions. Consequently, each parameter can condition the values of the others.

Suppose, as an example, that in a specific area there is a sudden seismic swarm of considerable intensity due to an intense surface fracturing phenomenon. The increase of this precursor involves a cascading effect on all the others; the fluids attract from the surrounding areas produce: (i) a greater intensity of outgassing; (ii) an increase in the concentration of radon gas; (iii) an increase in ground temperature due to upwelling of deep gases; (iv) a possible increase in ground deformation due to the action of surface fluids which reduce the mechanical strength of the rocks. All these correlated events produce interactions between the precursors, increasing their values.

To take these factors into account, statistical analyzes of past paroxysmal events were carried out in order to define, for each precursor, a threshold value for triggering the destructive event. Using specific Boolean functions, in which the main geophysical precursors and the respective threshold values appear, it is possible to take into account the contributions of each of them.

The results obtained for New Zealand, although restricted to only two precursors (seismic swarms and ground deformation) seem to confirm the initial hypothesis.

An effective solution was developed during the PhD work to advance the tools for environmental monitoring through satellite images. The GIS software system, implemented in Python, has been applied to different cities (Acireale, Kamakura and Ōhata) to qualitatively and quantitatively monitor the urban sprawl phenomenon. The development of specific algorithms that distinguish green and ur-

ban areas during the labelling phase and calculate geometric parameters (length, width and area) enabled us to process a swarm of images. The processing results confirmed the method's potential by simulating increases in green and urban areas in the hypothetical period between 2016 and 2020. The ability to trace the space-time evolution of specific portions of the territory in pre-set time intervals makes this innovative technique applicable to a wide range of natural phenomena.

The method application to a real case aims to test and refine the techniques previously used in the simulated case. The study of the variability of the heat areas in the periods concomitant to the two paroxysmal events of December 2015 and May 2016 was performed with a guided analysis to verify the consistency of the results.

Based on the INGV [116] press releases and the qualitative and quantitative feedback through Figure 4.6 and the Table 4.1, some considerations have been done. The heat fields show a marked variability attributable to three main cases: (i) before a paroxysmal event, thermal anomalies tend to extend considerably and move towards higher temperature classes; (ii) after the end of the event, a period of stasis with the heat cells that follow an opposite trend to the previous one (reduction of their surface and shift to lower temperature classes); (iii) in the event of violent paroxysmal activity of extraordinary intensity (December 2, 2015) there is a timely decrease in surface energy levels and an equally fast recharging phase signalled by a rapid rise in the intensity of the heat fields found in the image of December 19, 2015.

Ultimately, comparing the reports and the graphic and textual data confirmed the analysis's reliability using the GIS application.

Particular attention was also given to the code's readability and efficiency. Several parts of the GIS application have been optimized by refactoring the most computationally expensive portions to improve their execution times. Further restructuring phases have transformed the original monolithic architecture to the microservices one, adopting a Kafka-based streaming platform for data exchange between containers. Several portions of the code have been rewritten, adopting the

MapReduce instructions and the lambda function to improve the code's linearity and readability.

Code Snippets

```
countourCol="White"
while(actualCoordinate!=initialCoordinate) and (actualColor!=color["Red"])\
and (previousCoordinate!=actualCoordinate):
    previousCoordinate=actualCoordinate
    if img_1.getpixel((x+1,y+1))==color["Yellow"]:
        while img_1.getpixel((x+1,y+1))==color["Yellow"] and (actualCoordinate!=initialCoordinate)\
        and (actualColor!=color["Red"]):
            y+=1
            actualCoordinate=(x,y)
            actualColor=img_1.getpixel(actualCoordinate)
            lenghtBoundary+=1
            index+=1
            img_1.putpixel(actualCoordinate,color[countourCol])
            copyCoordAsJSON(boundaryCoord,actualCoordinate,number,index)
    elif img_1.getpixel((x,y+1))==color["Yellow"]:
        while img_1.getpixel((x,y+1))==color["Yellow"] and (actualCoordinate!=initialCoordinate)\
        and (actualColor!=color["Red"]):
            x+=1
            actualCoordinate=(x,y)
            actualColor=img_1.getpixel(actualCoordinate)
            lenghtBoundary+=1
            index+=1
            img_1.putpixel(actualCoordinate,color[countourCol])
            copyCoordAsJSON(boundaryCoord,actualCoordinate,number,index)
```

Figure A: The followBoundary algorithm.

```
1   def mergeImages(img,imgPoints,bndCoord,color,contour):
2       imgDraw=ImageDraw.Draw(img["main_image"])
3       dim=len(imgPoints["Image"]["Points"])
4       print("Load main image....")
5       for i in range(2,dim):
6           area,pair_1,pair_2=imgPoints["Image"]["Points"][f"p_{i}"]
7           drawTheme(area,color,pair_1,pair_2)
8       print("Load boundary....")
9       for k in range(1,len(bndCoord)):
10          for t in range(1,bndCoord[f"Boundary_{k}"]["NumberOfPoints"]+1):
11              drawBoundary(img,bndCoord,color,contour,k,t)
12
```

Figure B: The mergeImages algorithm.

```
def calcWidthAndLength(bC, X_MaxMin, Y_MaxMin, lst_XMin, lst_XMax, img, lstL, lstW, col, bMap,flag,val_1):
    print("Calculate the maximum and minimum value of x for each y.......")
    if flag==0:
        first=0
        dim=len(bC)
    else:
        first=val_1-1
        dim=val_1
    lst_1,lst_2,lst_3 = JsonHelper.initList()
    lst_MaxMin =[]
    for j in range(first,dim):
        JsonHelper.buildMaxMinXYAsJSON(bC, j)
        t = len(bC[f"Boundary_{j}"]["Points"])
        for i in range(t):
            points = bC[f"Boundary_{j}"]["Points"]
            lst = ((points[f"p_{i}"][0], points[f"p_{i}"][1]))
            if i == 0:
                lst_1 = [lst]
            else:
                lst_1.append(lst)
        lst_MaxMin = getValues(bC, lst_1, lst_MaxMin, X_MaxMin, Y_MaxMin, lst_XMin, lst_XMax, j, img,
                               lstL, lstW)
        JsonHelper.saveMaxMinXY(bC, lst_MaxMin, j)
        lst_2.append(lst_3)

```

Figure C: The calcWidthAndLength Algorithm

```
def defineGreenUrbanUnits(img,typeOfImage, bCoord, lstArea, col, flag, val_1):
    dim=0
    green = 0
    urban = 0
    if flag==0:
        first=0
        dim = len(bCoord)
    else:
        first=val_1-1
        dim=val_1
    typeOfArea = ""
    for k in range(first,dim):
        if flag!=1:
            JsonHelper.boundaryAreaPointsAsJSON(lstArea, k, 0)
        for m in range(len(bCoord[f"Boundary_{k}"]["X_max"])):
            x1,x2 = JsonHelper.setXMax(bCoord,k,m)
            x3,x4 = JsonHelper.setXMin(bCoord,k,m)
            if typeOfImage.getpixel((x1-1, x2)) == col["Green"]:
                green += 1
            elif typeOfImage.getpixel((x1-1, x2)) == col["Black"]:
                urban += 1
            if x3+1 < img["width"] and typeOfImage.getpixel((x3+1, x4)) == col["Green"]:
                green += 1
            elif x3+1 < img["width"] and typeOfImage.getpixel((x3+1, x4)) == col["Black"]:
                urban += 1
            if green > urban:
                typeOfArea = "Green_Area"
            else:
                typeOfArea = "Urban_Area"
        JsonHelper.setTypeOfArea(bCoord, lstArea, typeOfArea, k)
        green,urban = resetVariables(green,urban)

```

Figure D: The defineGreenUrbanUnits Algorithm

Algorithm 1 *Calc_Area* algorithm

Input: *bCoord* #Dictionary of the boundary coordinates
Output: *nothing*

Function Calc_Area():
  $pxArea \leftarrow 0$
  $colorArea \leftarrow "Green"$
  $n \leftarrow len(bCoord)$
  for $k = 1$ to n **do**
    $m \leftarrow len(bCoord[k][X_{min}])$
    if $bCoord[k]["Type\ of\ Area"] == "GreenArea"$ **then**
      $colorArea \leftarrow "Green"$
    else
      $colorArea \leftarrow "Black"$
    end if
    for $j = 1$ to m **do**
      $p_{min} \leftarrow bCoord[k][X_{min}][j]$
      $p_{max} \leftarrow bCoord[k][X_{max}][j]$
      $pxArea \leftarrow calcPxArea()$
    end for
    $bCoord[k]["Area(pixel)"] \leftarrow pxArea$
    $pxArea \leftarrow 0$
  end for

Algorithm 2 *calcPxArea* algorithm

Input: $colorArea, pxArea, p_{min}, p_{max}$
Output: $pxArea$

Function calcPxArea():
  $x_1 \leftarrow p_{min}[0]$
  $y_1 \leftarrow p_{min}[1]$
  $x_2 \leftarrow p_{max}[0]$
  for x_1 to x_2 **do** # $t = x_2 - x_1$
    if $getpixel(x_1 + 1, y_1) == colorArea$ **then**
      $pxArea \leftarrow pxArea + 1$
    end if
  end for
  return $pxArea$

Figure E: Pseudocode of the Calc_Area algorithm and the built-in calcPxArea function.

```
def calcXMaxOrMin(boundaryCoord,xType):
    print("Calculate",xType,".......")
    dim_1=len(boundaryCoord)
    for j in range(1,dim_1):
        k=1
        t=1
        boundaryCoord[f"Boundary_{j}"][xType]= {}
        dim_2=boundaryCoord[f"Boundary_{j}"]["NumberOfPoints"]
        for i in range(1,dim_2+1):
            x,y=setPoint(boundaryCoord,j,i)
            if i!=1:
                t=k
                if k>2:
                    while y!=boundaryCoord[f"Boundary_{j}"][xType][f"x_{t-1}"][1]:
                        t-=1
                        if t<=2:
                            break
                if (y==boundaryCoord[f"Boundary_{j}"][xType][f"x_{t-1}"][1]):
                    setXType(boundaryCoord,xType,j,t)
                else:
                    boundaryCoord[f"Boundary_{j}"][xType][f"x_{k}"]=(x,y)
                    k+=1
            else:
                boundaryCoord[f"Boundary_{j}"][xType][f"x_{t}"]=(x,y)
                k+=1

```

Figure F: The calcXMaxOrMin algorithm.

```
def windowMobile(img,x,y,color,range_x,range_y,sqArea):
    area=imageLabeling(img,x,y,color,range_x,range_y,sqArea)
    colorArea(x,y,color[area],img,"main_image",range_x,range_y)
    return area

def imageLabeling(image,x_1,y_1,col,r_x,r_y,squareArea):
    green=0
    black=0
    y_2=2
    x_2=2
    x_a,x_b=r_x
    y_a,y_b=r_y
    for x_2 in range(x_a,x_b):#x_a,x_b
      for y_2 in range(y_a,y_b):#y_a,y_b
        if x_2<image["width"] and y_2<image["height"]:
          green,black=pixelCount(image,x_2,y_2,green,black)
    area=calcPercent(image,x_2,y_2,squareArea,green,black)
    return area

```

Figure G: The windowMobile algorithm and its built-in imageLabeling function.

```
def calcWidth(bC,i,j,X_MinMax,maxV,minV,img,lstW,s):
    widthValues=0
    w=list(map(lambda x_1,x_2: (x_1[0],x_2[0],abs(x_1[0]-x_2[0]),x_1[1]), maxV,minV))
    w_lst=list(map(lambda x_3: x_3[2],w)) #list that contain all y
    max_W= max(w_lst)
    m=next((z for z, c in enumerate(w) if c[2] == max_W), None)
    '''(Xmin,Xmax,Width,y)'''
    widthValues=w[m]
    lstW.append((f"Boundary_{j}",max_W))
    X_MinMax.append([f"Boundary_{j}",(minV[0],maxV[len(maxV)-1])])
    JsonHelper.geometricParametersAsJSON(img,bC,i,j,widthValues)


def calcArea(bCoord,c1,img,col,endBoundary,lstX,px_area,lstArea,k):
    x_1=c1[0]
    if len(x_1)!=0:
        lstX.append(len(x_1))
    if endBoundary==True and len(lstX)!=0:
        px_area=reduce(lambda a,b: a+b,lstX)
        JsonHelper.setValuesArea(bCoord,lstArea,px_area,img,k)

```

Figure H: Functional implementation of the calcWidth and calcArea functions.

```
def jsonQuery(img,boundary_coord,color):
  query_1=True
  query_2=False
  for j in range(1,len(boundary_coord)):
     if query_1:
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Green_Area'\
        and boundary_coord[f"Boundary_{j}"]["Area"]["pixel"]<200:
            #area
            for i in range(1,len(boundary_coord[f"Boundary_{j}"]["X_max"])):
                x1= boundary_coord[f"Boundary_{j}"]["X_min"][f"x_{i}"]
                x2= boundary_coord[f"Boundary_{j}"]["X_max"][f"x_{i}"]
                img["main_image"].putpixel((x1,x2), color["Gray"])
            #boundary
            for k in range(1,len(boundary_coord[f"Boundary_{j}"]["Points"])):
                (x,y)=boundary_coord[f"Boundary_{j}"]["Points"][f"p_{k}"]
                img["main_image"].putpixel((x,y), color["Gray"])
     if query_2:
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Green_Area':
            area["Total_Green_Area"]+= boundary_coord[f"Boundary_{j}"]["Area"]["m^2"]
        if boundary_coord[f"Boundary_{j}"]["Area"]['Type of Area']=='Urban_Area':
            area["Total_Urban_Area"]+= boundary_coord[f"Boundary_{j}"]["Area"]["m^2"]
```

Figure I: A snapshot of some JSON queries

```
ENABLED = True
DISABLED = False

def runQueries(gCoord,col,input_path,output_path,w,h):
    listYears_1=["2017"]
    listYears_2=["2018","2019"]
    listYears_3=["2018","2019","2020"]
    listYears_4=["2017","2018","2019","2020"]
    filename=["2017", "2018-2019", "2018-2020", "2017-2020"]
    drawGradientData(listYears_1,filename[0],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_2,filename[1],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_3,filename[2],gCoord,col,w,h,output_path,ENABLED)
    drawGradientData(listYears_4,filename[3],gCoord,col,w,h,output_path,ENABLED)

def drawGradientData(listYears,fname,gCoord,col,w,h,out_path,flag):
    '''Colour all urban and green areas with different colours.'''
    if flag == DISABLED:
        return
    img=Image.open(f"{out_path}/background.png")
    fillColor=col["White"]
    for k in range(len(gCoord)):
        year=gCoord[f"Boundary_{k}"]["Date"]["year"]
        if year in listYears:
            fillColor=selectArea(gCoord,col,k)
            drawAreas(gCoord,img,col,fillColor,k)
            drawBoundaries(gCoord,img,col,fillColor,k)
    img.show()
    JsonHelper.saveImage(img,out_path,f"gradient_{fname}.png",'png')

```

Figure L: Queries obtained after swarm processing and applied to new areas.

List of Figures

List of Tables

Bibliography

[1] J.L. Hardebeck. "Spatial clustering of aftershocks impacts the performance of physics-based earthquake forecasting models." In: *Journal of Geophysical Research: Solid Earth* 126.2 (2021), e2020JB020824.

[2] U.S. Department of the Interior USGS. *Can you predict earthquakes?* URL: https://www.usgs.gov/faqs/can-you-predict-earthquakes.

[3] A. Bevilacqua, et al. "The Failure Forecast Method applied to the GPS and seismic data collected in the Campi Flegrei caldera (Italy) in 2011-2020." In: *Earth and Space Science Open Archive* (2021).

[4] C.R.J. Kilburn. "Forecasting Volcanic Eruptions: Beyond the Failure Forecast Method". In: *Front. Earth Sci.* 6 (2018).

[5] J. Vasseur et al. "Heterogeneity: The key to failure forecasting". In: *Scientific Reports* 5.1 (2015), p. 13259.

[6] A. Miyakoshi, M. Taniguchi, K. Ide, M. Kagabu, T. Hosono, J. Shimada. "Identification of changes in subsurface temperature and groundwater flow after the 2016 Kumamoto earthquake using long-term well temperature–depth profiles". In: *Journal of Hydrology* 582 (2020), pp. 1–10.

[7] D. Shanker, H.N. Singh, H. Paudyal, A. Kumar, A. Panthi, V.P. Singh. "Searching for an Earthquake Precursor—A Case Study of Precursory Swarm as a Real Seismic Pattern Before Major Shocks". In: *Pure and Applied Geophysics* 167 (2010), pp. 655–666.

[8] G. Bilotta, R. Spina, E. Tramontana. "Porting (Cuda/OpenCL) del software MagFlow per la simulazione dei flussi lavici dell'Etna [MagFlow porting software (Cuda/OpenCL) for simulation lava flows of Etna]". In: *Geologia tecnica ambientale* 2.14 (2014), pp. 33–71.

[9] E. Husni, K. Hamdi, and T. Mardiono. "Particle system implementation using smoothed Particle Hydrodynamics (SPH) for lava flow simulation." In: *2009 International Conference on Electrical Engineering and Informatics* 9 (2009), pp. 216–221.

[10] A. Hérault, G. Bilotta, R.A. Dalrymple. "SPH on GPU with CUDA". In: *Journal of Hydraulic Research* 48(Extra Issue) (2010), pp. 74–79.

[11] M. P. Johnson. "Environmental Impacts of Urban Sprawl: A Survey of the Literature and Proposed Research Agenda". In: *Environment and Planning A: Economy and Space* 33 (2001), pp. 717–35.

[12] T.J. Nechyba, R.P. Walsh. “Urban Sprawl”. In: *Journal of Economic Perspectives* 18 (2004), pp. 177–200.

[13] E. Patacchini et al. “Urban Sprawl in Europe”. In: *Brookings-Wharton Papers on Urban Affairs* 10 (2009), pp. 125–49.

[14] S. Habibi, N. Asadi. “Urban Sprawl in Europe”. In: *Procedia Engineering* 21 (2011), pp. 133–41.

[15] Z. Karakayaci. “The Concept of Urban Sprawling and Its Causes”. In: *Journal of International Social Research* 9 (2016).

[16] J. Dupras et al. “The Impacts of Urban Sprawl on Ecological Connectivity in the Montreal Metropolitan Region”. In: *Environmental Science Policy* 58 (2016), pp. 61–73.

[17] J. Dupras, M. Alam. “Urban Sprawling and Ecosystem Services: A Half-Century Perspective in the Montreal Region (Quebec, Canada)”. In: *Journal of Environmental Policy Planning* 17 (2015), pp. 180–200.

[18] J. Taylor, C. Paine, J. Fitzgibbon. “From Greenbelt to Greenways: Four Canadian Case Studies”. In: *Landscape and Urban Planning* 33 (1995), pp. 47–64.

[19] D.N. Bengston, J. Fletcher, K. Nelson. “Public Policies for Managing Urban Growth and Protecting Open Space: Policy Instruments and Lessons Learned in the United States”. In: *Landscape and Urban Planning* 69 (2004), pp. 271–86.

[20] J.C. Cooper. “Legislation to Protect and Replace Trees on Private Land: Ordinances in Westchester County, New York”. In: *Journal of Arboriculture* 22 (1996), pp. 270–8.

[21] J. Houde. “Public Property Tree Preservation’, Journal of Arboriculture”. In: *Journal of Arboriculture* 23 (1997), pp. 83–6.

[22] R. Spina, E. Tramontana. “An Image-Processing Approach for Computing the Size of Green Areas in Cities”. In: *The 2021 9th International Conference on Computer and Communications Management (ICCCM), Singapore* (2021).

[23] M. Krzysztoń, B. Śnieżyński. “Combining machine learning and multi-agent approach for controlling traffic at intersections”. In: *Computational Collective Intelligence, LNCS* 9329 (2015), pp. 57–66.

[24] D. Grzonka, A. Jakóbik, J. Kołodziej, S. Pllana. “Using a multi-agent system and artificial intelligence for monitoring and improving the cloud performance and security”. In: *Future Generation Computer Systems, Elsevier* 86 (2018), pp. 1106–1117.

[25] A. E. F. Seghrouchni, A. M. Florea, A. Olaru. “Multi-agent systems: A paradigm to design ambient intelligent applications”. In: *Studies in Computational Intelligence, SCI* 315 (2010), pp. 3–9.

[26] V. Suryanarayanan, G. Theodoropoulos, M. Lees. "PDES-MAS: Distributed simulation of multi-agent systems". In: *Procedia Computer Science, Elsevier* 18 (2013), pp. 671–681.

[27] A. Calvagna, E. Tramontana. "Delivering dependable reusable components by expressing and enforcing design decisions". In: *Proc. of IEEE Computer Software and Applications Conference (COMPSAC), IEEE* (2013), pp. 493–498.

[28] M. Ripepe et al. "Infrasonic Early Warning System for Explosive Eruptions". In: *JGR Solid Earth* 123.118 (2018), pp. 9570–9585.

[29] G. Cua, T. Heaton. "The Virtual Seismologist (VS) Method: a Bayesian Approach to Earthquake Early Warning". In: *Earthquake Early Warning System, Springer, Berlin, Heidelberg* (2007).

[30] M. Böse, T. Heaton, E. Hauksson. "Rapid Estimation of Earthquake Source and Ground-Motion Parameters for Earthquake Early Warning Using Data from a Single Three-Component Broadband or Strong-Motion Sensor". In: *GeoScienceWorld* 102.2 (2012), pp. 738–750.

[31] R. Spina, E. Tramontana. "GAP: a software system for the analysis of geological data". In: *Geologia Tecnica Ambientale* 4 (2010), pp. 15–42.

[32] Y.H. Oh, G. Kim. "A radon-thoron isotope pair as a reliable earthquake precursor". In: *Scientific Reports* 5 (2015), pp. 1–6.

[33] D. Coppola et al. "Thermal Remote Sensing for Global Volcano Monitoring: Experiences From the MIROVA System". In: *Frontiers in Earth Science* 7 (2020).

[34] M.E. West" "H. Buurman. "Seismic Precursors to Volcanic Explosions During the 2006 Eruption of Augustine". In: *Conference Proceedings* (2010).

[35] M. Moro et al. "New insights into earthquake precursors from InSAR". In: *Scientific Reports* 7.1 (2017), pp. 1–11.

[36] C. Cavallaro, G. Verga, E. Tramontana, O. Muscato. "Suggesting just enough (un)crowded routes and destinations". In: *Proc. of 21st Workshop 'From Objects to Agents' (WOA 2020), Bologna, Italy* (2020), pp. 493–498.

[37] G. Bunea, F. Leon, G. Atanasiu. "Postdisaster evacuation scenarios using multiagent system". In: *Journal of Computing in Civil Engineering, American Society of Civil Engineers* (2016), p. 30.

[38] F. Fiedrichn. "An hla based multiagent system for optimized resource allocation after strong earthquakes". In: *J. Cohen (Ed.), Proceedings of the 2006 Winter Simulation Conference, Monterey, CA* (2006), pp. 486–492.

[39] D. Moser, D. Pinto, A. Cipriano. "Developing a multiagent based decision support system for realtime multi-risk disaster management". In: *International Journal of Environmental and Ecological Engineering* 9 (2015), pp. 831–835.

[40] R. Spina, A. Fornaia, E. Tramontana. “VSEW: an early warning system for volcanic and seismic events”. In: *Proceedings of IEEE International Conference on Smart Computing, SMARTCOMP, Bologna, Italy* (2020).

[41] A. Tani, T. Yamamura, Y. Waridashi, H. Kawamura, A. Takizawa. “Simulation on rescue in case of earthquake disaster by multi-agent system”. In: *Proc. of World Conference on Earthquake Engineering, Vancouver, BC, Canada* (2004), pp. 1–6.

[42] G. Bunea, G. M. Atanasiu, F. Leon. “The effect of information on the performance of a multiagent system for post-disaster evacuation”. In: *Proc. of International Symposium on Life-Cycle Civil Engineering, Delft, The Netherlands* (2016), pp. 2053–2059.

[43] T. Furuya, S. Sadohara. “Modeling and simulation of rescue activity by the local residents in the seismic disaster”. In: *Proc. of ESRI International User Conference* (2004).

[44] I.G.Main, A.F.Bell, P.G.Meredith, S.Geiger, S.Touati. “The dilatancy-diffusion hypothesis and earthquake predictability”. In: *Geological Society, London, Special Publications* 367 (2012), pp. 215–230.

[45] K. Aki. “Asperities, barriers, characteristic earthquakes and strong motion prediction”. In: *Journal of Geophysical Research* 89 (1984), pp. 5867–587.

[46] M. Béjar-Pizarro, et al. “Asperities and barriers on the seismogenic zone in north chile: State of the art after the 2007 mw 7.7 tocopilla earthquake inferred by GPS and InSar data”. In: *Geophysical Journal International* 183 (2010), pp. 390–406.

[47] S. Awadallah, D. Moure, P. Torres-González. “An internet of things (IoT) application on volcano monitoring”. In: *Sensors 19* 4651 (2019).

[48] N. Bajo, A. Di Noi. “Reti ecologiche e paesaggi metropolitani”. In: *II Rapporto APAT “Qualità dell’ambiente urbano”* (2005).

[49] D.J. Nowak. “Trees pollute? A “TREE” explains it all”. In: *Proc. of National Urban Forest Conference, American Forests, Washington DC”* (1995).

[50] D.J. Nowak, P.J.McHale, M. Ibarra, D. Crane, J. Stevens, C. Luley. “Modelling the effects of urban vegetation on air pollution”. In: *Air Pollution Modelling and Its Application XII Plenum Press, New York”* (1998), pp. 399–407.

[51] L. Tyrvainen, S. Paulei, K. Seeland, S. De Vries. “Benefits and Uses of urban forests and trees”. In: *Urban forests and trees in Europe: A Reference Book,Springer Verlag* (2005), pp. 81–114.

[52] C. Cengiz, B. Cengiz, P. Keçecioğlu Dağlı. “The Role Of Urban Green Areas On Social Sustainability: Bartın Case Study”. In: *Proceedings of International Multidisciplinary Congress of Eurasia, Barcelona* (2018).

[53] C. Alexander, S. Ishikawa, M. Silverstein. "A Pattern Language: Towns, Buildings, Construction." In: *Proceedings of International Multidisciplinary Congress of Eurasia, Barcelona, Oxford University Press, USA* (1977).

[54] G. Meera Gandhi, S. Parthiban, N. Thummalu, A. Christy. "Ndvi: Vegetation change detection using remote sensing and gis – A case study of Vellore District." In: *Elsevier ICRTC, Procedia Computer Science,* 57 (2015), pp. 1199–1210.

[55] S. Lang, et al. "Quantifying and Qualifying Urban Green by Integrating Remote Sensing, GIS, and Social Science Method." In: *Use of Landscape Sciences for the Assessment of Environmental Security, Springer Netherland* (2008), pp. 93–105.

[56] K. Anderson, S. Hancock, S. Casalegno, A. Griths, D. Griths, F. Sargent, J. McCallum, D.T.C. Cox, K.J. Gaston. "Visualising the urban green volume: Exploring LiDAR voxels with tangible technologies and virtual models." In: *Landscape and Urban Planning* 178 (2018), pp. 248–260.

[57] A. Kundu, D.K. Dutta. "Monitoring desertification Risk through Climate Change and Human Interference Using Remote Sensing and GIS Techniques." In: *International Journal of Geomatics and Geosciences* 2.1 (2011).

[58] F. Djeddaoui, M. Chadli, R.Gloaguen, "Desertification Susceptibility Mapping Using Logistic Regression Analysis in the Djelfa Area, Algeria." In: *Remote Sensing* 9.10 (2017).

[59] Q. Guo, B. Fu, P. Shi, T. Cudahy, J. Zhang, H. Xu. "Satellite Monitoring the Spatial-Temporal Dynamics of Desertification in Response to Climate Change and Human Activities across the Ordos Plateau, China." In: *Remote Sensing* 9.6 (2017).

[60] F. Rubiera-Morollón, R. Garrido-Yserte. "Recent Literature about Urban Sprawl: A Renewed Relevance of the Phenomenon from the Perspective of Environmental Sustainability." In: *Sustainability* 12.16 (2020).

[61] H. Hashim, Z.A. Latif, N.A.Adnan. "Urban Vegetation Classification with NDVI Threshold Value Method with Very High Resolution (VHR) Plaiades Imagery." In: *Proc. of Intern. Conference on Geomatics and Geospatial Technology (GGT), Kuala Lumpur, Malaysia.* (2019).

[62] S. Lahoti, M., Kefi, A., Lahoti, O., Saito. "Mapping Methodology of Public Urban Green Spaces Using GIS: An Example of Nagpur City, India." In: *Sustainability* 11.7 (2019).

[63] A.F. de Castro, T. Martin. "Fuzzy Logic Applied Spatial Queries in Geographic Information Systems." In: *Journal of Computer Science* (2018).

[64] K.M. Atikur Rahman, D. Zhang. "Analyzing the Level of Accessibility of Public Urban Green Spaces to Different Socially Vulnerable Groups of People." In: *Sustainability* 10.11 (2018), pp. 1–27.

[65] Sudarshan Madabusi and Suryakanth V. Gangashetty. "Edge detection for facial images under noisy conditions". In: *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*. 2012, pp. 2689–2693.

[66] Yingkui L., Charles M. O., and G. Yonggui. "GIS-based detection of grain boundaries". In: *Journal of Structural Geology* 30.4 (2008), pp. 431–443.

[67] Atul M. Gonsai Anil K.B. "An improved edge detection algorithm for X-Ray images based on the statistical range". In: *Heliyon* 5.10 (2019).

[68] H. Song, P.H. Nguyen, F. Chauvel, J. Glattetre, T. Schjerpen. "Customizing MultiTenant SaaS by Microservices: A Reference Architecture." In: *IEEE ICWS 2019 At: Milan, Italy* (2019).

[69] P Mohan et al. "Analysis on Fingerprint Extraction Using Edge detection and Minutiae Extraction". In: *2019 2nd International Conference on Signal Processing and Communication (ICSPC)*. 2019, pp. 161–164. DOI: 10.1109/ICSPC46172.2019.8976803.

[70] Pawan Sinha et al. "Face Recognition by Humans: Nineteen Results All Computer Vision Researchers Should Know About". In: *Proceedings of the IEEE* 94.11 (2006), pp. 1948–1962. DOI: 10.1109/JPROC.2006.884093.

[71] Jason Brownlee. *Deep learning for computer vision: image classification, object detection, and face recognition in python*. Machine Learning Mastery, 2019.

[72] Dan Ciregan, Ueli Meier, and Jürgen Schmidhuber. "Multi-column deep neural networks for image classification". In: *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012, pp. 3642–3649.

[73] Rafał Scherer. *Computer vision methods for fast image classification and retrieval*. Springer, 2020.

[74] Shenghua Gao, Ivor Wai-Hung Tsang, and Liang-Tien Chia. "Kernel sparse representation for image classification and face recognition". In: *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer. 2010, pp. 1–14.

[75] P Apoorva et al. "Automated criminal identification by face recognition using open computer vision classifiers". In: *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*. IEEE. 2019, pp. 775–778.

[76] J Harikrishnan et al. "Vision-face recognition attendance monitoring system for surveillance using deep learning technology and computer vision". In: *2019 international conference on vision towards emerging trends in communication and networking (ViTECoN)*. IEEE. 2019, pp. 1–5.

[77] Hongying Meng, Nick Pears, and Chris Bailey. "A human action recognition system for embedded computer vision application". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–6.

[78] Mahmoud Al-Faris et al. "A review on computer vision-based methods for human action recognition". In: *Journal of imaging* 6.6 (2020), p. 46.

[79] H. Wang, Y. Ni, L. Sun, Y. Chen, T. Xu, X. Chen, W. Su, Z. Zhou. "Hierarchical visualization of geographical areal data with spatial attribute association." In: *Visual Informatics* 5 (2021), pp. 89–91.

[80] Debalina Barik and Manik Mondal. "Object identification for computer vision using image segmentation". In: *2010 2nd International conference on education technology and computer*. Vol. 2. IEEE. 2010, pp. V2–170.

[81] Anastasia Sofou, Georgios Evangelopoulos, and Petros Maragos. "Soil image segmentation and texture analysis: a computer vision approach". In: *IEEE Geoscience and Remote Sensing Letters* 2.4 (2005), pp. 394–398.

[82] Rajesh S Sarkate, NV Kalyankar, and PB Khanale. "Application of computer vision and color image segmentation for yield prediction precision". In: *2013 International Conference on Information Systems and Computer Networks*. IEEE. 2013, pp. 9–13.

[83] Yu-Ho Tseng and Shau-Shiun Jan. "Combination of computer vision detection and segmentation for autonomous driving". In: *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE. 2018, pp. 1047–1052.

[84] D. Stratoulias, V. Tolpekin, R.A. De By, R. Zurita-Milla, V. Retsios, W. Bijker, M.A. Hasan, E. Vermote. "A Workflow for Automated Satellite Image Processing: from Raw VHSR Data to Object-Based Spectral Information for Smallholder Agriculture." In: *Remote Sensing* 9.10 (2017).

[85] K. Ostir, K. Cotar, A. Marsetic, P. Pehani, M. Perse, K. Zaksek, J. Zaletelj, T. Rodic. "Automatic Near-Real-Time Image Processing Chain for Very High Resolution Optical Satellite Data." In: *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XL7 (2015), pp. 669–676.

[86] W. Nina, R. Cruz, J. Serrano, J. Cuba, Y. Huaynacho, A. Mamani-Aliaga, Y. Yari, P. Yanyachi. "A new approach to the massive processing of satellite images." In: *Latin American Computing Conference (CLEI)* XL7 (2015), pp. 1–6.

[87] H. Deng, S. Huang, Q. Wang, Z. Pan, Y. Xin. "Building high-performance system for processing a daily large volume of chinese satellites imagery." In: *High-Performance Computing in Remote Sensing IV* 9247 (2014).

[88] X. Zhang. "Micro-service architecture in reconstruction and optimization of gis application service." In: *Chemical Engineering Transactions* 66 (2018), pp. 847–852.

[89] J.M. Ibáñez, et al. “TOMO-ETNA experiment at Etna volcano: activities on land.” In: *Annals of Geophysics* 59.4 (2016).

[90] S. Petrosino, C. Ricco, I. Aquino. “Modulation of Ground Deformation and Earthquakes by Rainfall at Vesuvius and Campi Flegrei (Italy).” In: *Frontiers in Earth Science* 9 (2021).

[91] R. Bernini, R. Gravina, A. Minardo, L. Zeni, Z. Petrillo, M. Piochi, R. Scarpa. “Long-Term Temperature Monitoring of Volcanic Areas by Distributed Optical Fiber Sensors”. In: *Sensors and Microsystems* (2008), pp. 345–349.

[92] L.M. Peci, M. Berrocoso, A. Fernández-Ros, A. García, J.M. Marrero, R. Ortiz. “Embedded ARM System for Volcano Monitoring in Remote Areas: Application to the Active Volcano on Deception Island (Antarctica)”. In: *Sensors* 14 (2014), pp. 672–690.

[93] H. Askarinejad, A. Chakraborty, I. Williamson. “Application of IoT-based systems in seismic monitoring of structures.” In: *New Zealand Society for Earthquake Engineering (NZSEE)At: Auckland, NZ* (2018).

[94] L. Lin, H. Yang, X. Xu. “Effects of Water Pollution on Human Health and Disease Heterogeneity: A Review”. In: *Front. Environ. Sci.* 10 (2022).

[95] J. Halder, N. Islam, N. Islam. “Water Pollution and its Impact on the Human Health.” In: *Eh 2* 1 (2015), pp. 36–46.

[96] M.P. Choudhary, V. Garg. “Causes, Consequences and Control of Air Pollution.” In: *Conference: All India Seminar on Methodologies for Air Pollution.* (2013).

[97] S. Dixit, S. Vasupradha, S.V. Todurkar, K.S. Harshitha, O. Alekhya. “Air Pollution Control Using IoT: A Survey.” In: *Proceedings of Third International Conference on Intelligent Computing, Information and Control System.* (2022), pp. 585–599.

[98] D. Lingvay, A. Bors, A.G. Bors. “Electromagnetic pollution and its effects on living matter.” In: *Electrotehnică, Electronică, Automatică.* 66.2 (2018), pp. 5–11.

[99] I. Lingvay, A. Voina, C. Lingvay, C. Mateescu. “The impact of the electromagnetic pollution of theenvironment on the complex build-up media.” In: *Revue Roumaine des Sciences Techniques, série Électrotechniqueet Énergétique.* 53.2bis (2008), pp. 95–112.

[100] H. JariwalaHiral, S. SyedMinarva, J. PandyaYogesh, M. Gajera. “Noise Pollution Human Health: A Review.” In: *Conference: Noise and Air Pollution: Challenges and Opportunities.* 53.2bis (2017), pp. 95–112.

[101] Z.U.R. Farooqi, M. Sabir, J. Latif, P. Ilić. “Assessment of noise pollution and its effects on human health in industrial hub of Pakistan.” In: *Environmental Science and Pollution Researc.* 27.3 (2020), pp. 1–10.

[102] S. Fabre, R. Gimenez, A. Elger, T. Rivière. "Unsupervised Monitoring Vegetation after the Closure of an Ore Processing Site with Multi-Temporal Optical Remote Sensing." In: *Sensors* 20.17 (2020).

[103] G. Zhao, Y. Zhang, J. Tan, C. Li, Y. Ren. "A Data Fusion Modeling Framework for Retrieval of Land Surface Temperature from Landsat-8 and MODIS Data." In: *Sensors* 20.15 (2020).

[104] K. Li, M. Feng, A. Biswas, H. Su, Y. Niu, J. Cao. "Driving Factors and Future Prediction of Land Use and Cover Change Based on Satellite Remote Sensing Data by the LCM Model: A Case Study from Gansu Province, China." In: *Sensors* 20.10 (2020).

[105] J.J. Alonso del Rosario, J.M. Vidal Pérez, E. Blázquez Gómez. "On the Prediction of Upwelling Events at the Colombian Caribbean Coasts from Modis-SST Imagery." In: *Sensors* 19.13 (2019).

[106] H. Qi, F. Huang, H. Zhai. "Monitoring Spatio-Temporal Changes of Terrestrial Ecosystem Soil Water Use Efficiency in Northeast China Using Time Series Remote Sensing Data." In: *Sensors* 19.6 (2019).

[107] N. Pasichnyk, D. Komarchuk, O. Opryshko, S. Shvorov, V. Reshetiuk and B. Oksana. "Technologies for Environmental Monitoring of the City." In: *2021 IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)* (2021), pp. 40–43.

[108] F. Eugenio, J. Martin, J. Marcello, E. Fraile-Nuez. "Environmental monitoring of El Hierro Island submarine volcano, by combining low and high resolution satellite imagery." In: *International Journal of Applied Earth Observation and Geoinformation* 29 (2014), pp. 53–66.

[109] D.D. Alexakis, A. Agapiou, M. Tzouvaras, et al. "Integrated use of GIS and remote sensing for monitoring landslides in transportation pavements: the case study of Paphos area in Cyprus." In: *Nat Hazards* 72 (2014), pp. 119–141.

[110] E. Tamassoki, H. Amiri, Z. Soleymani. "Monitoring of shoreline changes using remote sensing (case study: coastal city of Bandar Abbas)." In: *IOP Conference Series: Earth and Environmental Science* 20.1 (2014), p. 012023.

[111] Y.A. Papadimitrakis. "Monitoring water quality in water supply and distribution systems". In: *Advances in Water Supply Management (C. Maksimovic, D. Butler, and FA Memon, eds.)* (2003), pp. 441–450.

[112] I. Manfredonia et al. "An early-warning aerospace system for relevant water bodies monitoring". In: *2015 IEEE Metrology for Aerospace (MetroAeroSpace)*. 2015, pp. 536–540. DOI: 10.1109/MetroAeroSpace.2015.7180714.

[113] O. Ivanik et al. "Geological and geomorphological factors of natural hazards in Ukrainian Carpathians". In: *Journal of Ecological Engineering* 20.4 (2019).

[114] Diofantos Hadjimitsis et al. “Exploring natural and anthropogenic risk for cultural heritage in Cyprus using remote sensing and GIS”. In: *International Journal of Digital Earth* 6.2 (2013), pp. 115–142.

[115] F. Cigna et al. “Ground instability in the old town of Agrigento (Italy) depicted by on-site investigations and Persistent Scatterers data”. In: *Natural Hazards and Earth System Sciences* 12.12 (2012), pp. 3589–3603.

[116] INGV Catania. *Aggiornamenti sull'attività eruttiva dell'Etna (dall'INGV-Osservatorio Etneo, Catania)*. URL: http://www2.ct.ingv.it/en/component/content/article/102-aggiornamenti-etna/889-aggiornamenti-etna.html.

[117] USGS. *Using the USGS Landsat Level-1 Data Product*. URL: https://www.usgs.gov/landsat-missions/using-usgs-landsat-level-1-data-product.

[118] Q. Weng, D. Lu, and J. Schubring. “Estimation of land surface temperature–vegetation abundance relationship for urban heat island studies.” In: *Remote Sensing of Environment* 89.4 (2004), pp. 467–483.

[119] M. Ahmad and Dr.Ghulam Rasul. “Prediction of Soil Temperature by Air Temperature; a Case Study for Faisalabad.” In: *Pakistan Journal of Meteorology* 5 (2008).

[120] 3BMeteo. *Dati meteo storici*. URL: https://www.3bmeteo.com/meteo/vulcano/storico/201805.

[121] G. Graci, P. Pileri, and M. Sedazzari. “Guida all'uso di ArcGIS per l'analisi del territorio e la valutazione ambientale”. In: *Collana GIS, Dario Flaccovio editore* (2009), pp. 1–270.

[122] N. Sabi'u, S.N. Muhammed, N. Zakari, M.S. Khalil. “Vector data model in GIS and how it underpins a range of widely used spatial analysis technique.” In: *Dutse Journal of Pure and Applied Sciences* 1.1 (2020), pp. 3023–3049.

[123] E.A. Haller. “Geospatial Analysis Framework.” In: *Brain. Broad Research in Artificial Intelligence and Neuroscience* 1.2 (2010), pp. 166–171.

[124] R. Spina, E. Tramontana. “An automated classification system for urban areas matching the “City Country Fingers” pattern: the cases of Kamakura (Japan) and Acireale (Italy) cities”. In: *Journal of Urban Ecology, Oxford University Press* 7.1 (2021).

[125] Franch-Pardo, B.M. Napoletano, F. Rosete-Verges, L. Billa. “A review. Science of The Total Environment”. In: *Spatial analysis and GIS in the study of COVID-19* 739 (2020).

[126] A. Comber, C. Brundson, E. Green. “Using a GIS-based network analysis to determine urban greenspace accessibility for different ethnic and religious groups.” In: *Landscape and Urban Planning* 86.1 (2008), pp. 103–114.

[127] M. El Meseery, O. Hoeber. "Geo-Coordinated Parallel Coordinates (GCPC): Field trial studies of environmental data analysis". In: *Visual Informatic* 2.2 (2018), pp. 111–124.

[128] D. La Rosa. "Accessibility to greenspaces: GIS based indicators for sustainable planning in a dense urban context." In: *Ecological Indicators* 42 (2014), pp. 122–134.

[129] M.O. Arnous, Y.M. Sultan. "Geospatial technology and structural analysis for geological mapping and tectonic evolution of Feiran–Solaf metamorphic complex, South Sinai, Egypt". In: *Arabian Journal of Geosciences* 7.8 (2014), pp. 3023–3049.

[130] R. Vatseva, M. Kopecká, J. Otahel, K. Rosina, A. Kitev, S.E. Genchev. "Mapping Urban Green Spaces Based On Remote Sensing Data: Case Studies in Bulgaria and Slovakia". In: *Proc. of Intern. Conference on Cartography and GIS, Albena (Bulgaria)* (2016).

[131] G. Cillis, D. Statuto, P. Picuno. "Vernacular Farm Buildings and Rural Landscape: A Geospatial Approach for Their Integrated Management." In: *Sustainability* 12.1 (2020).

[132] A.A. Devendran, G. Lakshmanan. "A review on accuracy and uncertainty of spatial data and analyses with special reference to urban and hydrological modelling". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* II.8 (2014), pp. 156–176.

[133] Y.C. Sae-Jung, P. Do Minh Xiao. "Error propagation modeling in gis polygon overlay". In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XXXVII. Part B2 (2008).

[134] C. Brunsdon, S. Openshaw. "Simulating the Effects of Error in GIS". In: *Geographic Information Handling: Research and applications* (1993).

[135] F. Biljecki, G.B.M Heuvelink, H. Ledoux, J. Stoter. "The effect of acquisition error and level of detail on the accuracy of spatial analyses." In: *Cartography and Geographic Information Science* 45.2 (2018), pp. 156–176.

[136] G.M.B. Heuvelink, P.A. Burrough. "Propagation of Errors in Spatial Modeling with GIS." In: *International Journal of Geographic Information System* 3.4 (2002).

[137] T.G. Wade, J.D. Wickham, M.S. Nash, A.C. Neale, K.H. Riitters, K.B. Jones. "From Paper Map to Geospatial Vector Layer." In: *IASSIST quarterly/International Association for Social Science Information Service and Technology* 42.3 (2018), pp. 1–24.

[138] R.G. Congalton. "Exploring and Evaluating the Consequences of Vector-to-Raster and Raster-to-Vector Conversion." In: *Photogrammetric Engineering and Remote Sensing* 63.4 (1997), pp. 425–434.

[139] J. ChenC, H. Zhou, M. Cheng. "Area error analysis of vector to raster conversion of areal feature in GIS." In: *Acta Geodaetica et Cartographica Sinica* 36.3 (2007), pp. 344–350.

[140] S. Liao, Z. Bai, Y. Bai. "Errors prediction for vector-to-raster conversion based on map load and cell size." In: *Chinese Geographical Science* 22 (2012), pp. 695–704.

[141] X. Hilaire, K. Tombre. "Robust and Accurate Vectorization of Line Drawings." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.6 (2006), pp. 890–904.

[142] V. Lacroix. "Raster-to-Vector Conversion: Problems and Tools Towards a Solution a Map Segmentation Application." In: *In IEEE Xplore. Seventh International Conference on Advances in Pattern Recognition* (2009).

[143] K. Gos, W. Zabierowski. "The Comparison of Microservice and Monolithic Architecture". In: *IEEE XVIth International Conference on the Perspective Technologies and Methods in MEMS Design (MEMSTECH)* (2020), pp. 150–153.

[144] S. Barakat. "Monitoring and Analysis of Microservices Performance." In: *Journal of Computer Science and Control Systems* 10.1 (2017), pp. 19–22.

[145] G. Steinacker, W. Hasselbring. "Microservice Architectures for Scalability, Agility and Reliability in E-Commerce". In: *Proceedings 2017 IEEE International Conference on Software Architecture Workshops* (2017).

[146] M. Baboi, A. Iftene, D. Gifu. "Dynamic Microservices to Create Scalable and Fault Tolerance Architecture." In: *Procedia Computer Science* 159 (2019), pp. 1035–1044.

[147] G. Cherradi, A. El Bouziri, A. Boulmakoul, K. Zeitouni. "Real-time azMat Environmental Information System: A micro-service based architecture." In: *Procedia Computer Science* 109C (2017), pp. 982–987.

[148] A.V. Vorob'evab, V.A. Pilipenkocb, T.A. Enikeeva, G.R. Vorobevaa. "Geoinformation system for analyzing the dynamics of extreme geomagnetic disturbances from observations of ground stations." In: *Computer Optics* 44.5 (2020), pp. 782–790.

[149] A. Krylovskiy, M. Jahn, E. Patti. "Designing a smart city internet of things platform with microservice architecture." In: *3rd International Conference on Future Internet of Things and Cloud* (2015), pp. 25–30.

[150] M. Blockhuys. *Microservices: a paradigm shift for fast growing e-commerce businesses.* URL: https://www.sqli.nl/en/blog/whitepaper/microservices-fast-growing-e-%20commerce.