POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

Visibility analysis in strategic scenario using innovative GIS tools

Supervisors

Prof. Marco PIRAS

Candidate

Raffaele PEZONE

ACADEMIC-YEAR 2022/2023

Summary

In the recent strategic scenery and military operations, it is important to have a good comprehension of employed areas in way to work in safe and well-placed environments. The big challenge is to find a safe zone to stay in, well protected by territorial dominance. The safeness of a strategic stakeout depends on the visual of the surrounding environment and the level of self-coverage. The attention must concentrate also on the streets, which are the crucial point of movement. Having a padronance of the surrounding streets means having control of the entire zone. The key point is the continuous usage of traditional methods based on digital and paper geographical maps devoted to finding the best zone to use. Those traditional methods require a lot in terms of resources and time and don't allow for a tridimensional view of the problem. This thesis proposes the development of a dynamic visibility analysis tool for various scenarios in a strategic stakeout, with the aim of improving comprehension of its effectiveness and critical points. The proposed approach involves developing a Desk tool and a Mobile tool based on GIS and Python adopting a DEM model. The tools fulfill the necessity of having a useful instrument able to analyze a considered area from a chosen observation point or an established observation zone. The analysis is both on the surrounding area and streets to allow a well-developed or rapid response to an analysis request in safe or dangerous situations. The input data include a Digital Terrain Model or a Digital Surface Model derived from Open Data or generated through aerial surveys using an unmanned aerial vehicle (UAV) and a street map of the considered area. This research enables the analysis of the safety of a strategic stakeout in different scenarios, leading to improved decision-making and enhanced safety measures. The usage of analysis tools will open the possibility of a new typology of tests and simulations conducted on the field. Strategically it could be very useful to simulate the effectiveness of different real strategic stakeouts and simulate the response in dangerous conditions.

Acknowledgements

"A tutti quelli che ci sono sempre stati e che ci saranno, grazie"
Raffaele

Table of Contents

List of Tables

List of Figures

Acronyms

GIS
Geographic Information System

DTM
Digital Terrain Model

DSM
Digital Surface Model

POA
Posto Osservazione e Allarme

CIGC
Centro Interforze di Gestione e Controllo

IGM
Istituto Geografico Militare

DEM
Digital Elevation Model

DDEM
Dense Digital Elevation Model

DDTM
Dense Digital Terrain Model

DDSM
Dense Digital Surface Model

IIM

Istituto Idrografico della Marina

CIGA

Centro Informazioni Geotopografiche Aeronautiche

UTM

Universal Transverse of Mercator

CTR

Carta Tecnica Regionale

SINT

Sistema Informativo Nazionale per la Gestione e la Conservazione del Territorio e dell'Ambiente

OSM

OpenStreetMap

TIN

Triangulated Irregular Network

TIFF

Tagged Image File Format

KML

Keyhole Markup Language

IS

Information System

GNU

General Public License

SQL

Structured Query Language

API

Application Programming Interface

GUI

Graphical User Interface

WMS

Web Map Service

WFS

Web Feature Service

WCS

Web Coverage Service

SWE

Sensor Web Enablement

CS-W

Catalog Services - Web

GSW

Geospatial Semantic Web

SSW

Semantic Sensor Web

CC BY

Creative Commons Attribution

OSGeo

Open Source Geospatial Foundation

REST

Representational state transfer

CORS

Cross-origin resource sharing

Introduction

In the recent strategic scenery, military operation duties are evolving simultaneously with new technologies available on the market. Technology's power plays the main role in satisfying everyday challenges. Situational awareness and a deep understanding of the operational theater are fundamental to reducing the risk of failure. In an era characterized by complex geopolitical dynamics, decision-making depends on acquiring knowledge about the terrain, infrastructure, and potentially dangerous situations. The success of an operation is influenced by the ability to perceive, analyze, and exploit the spatial attributes of the location. Finding a good position for a strategic outpost is the primary task, followed by self-adaptation and planning in every situation. Effective mission planning is essential for mission success, as it helps anticipate and analyze potential risks well in advance, thereby avoiding them. The temporal dimension is also a critical factor. Time is a limited resource but fundamental in obtaining useful information about the operational area. The worst situation for an operation is to make important decisions about the area without information. The risks are associated with the potential consequences of inadequate planning, which could lead to discovering or detecting certain missions in dangerous environments, posing potential threats. Moreover, road network awareness in and around the operational area increases his full control and surveillance. Roads are the lifeblood of movement and logistics in any operation, and his block is one of the biggest elements for the operation failure. Therefore, the planning phase for strategic operations is fundamental and requires a lot of time and resources, heavily relying on topographic maps, satellite images, and traditional methodologies. Such extended planning periods can be an obstruction nowadays, where rapid decision-making is the key. In particular, operational problems may arise concerning planning time, environmental safety or danger, and communication availability. The planning process often takes a considerable amount of time, and in certain situations, end users may face time constraints for conducting field reconnaissance. The difference between gaining control of an area and achieving a competitive advantage on the field can be a matter of days or even hours. Urgent situations may not allow extensive field surveys or map and satellite image study days. In both safe and dangerous scenarios, time is critical, and the ability to

adapt quickly to changing circumstances is paramount. Effective communication with remotely monitoring personnel is essential, especially in connectivity-limited environments. Online or direct connections with remote monitoring personnel are not always available, emphasizing the vital role of technology in enhancing military responsiveness and preparedness. The Geographic Information Systems (GIS) tools have emerged as useful assets in this challenging and dynamic environment. In the phase of operation study, innovative GIS tools can revolutionize the way to approach strategic planning and operational execution by enabling a comprehensive understanding of the field. They offer the potential to optimize the temporal aspect of planning, exploiting the power of GIS technology and reducing the time required for data collection and analysis. This translates to a competitive advantage in the modern theater of operations and the technology sector. Additionally, GIS tools facilitate efficient route planning and logistics management. The road network can be analyzed and optimized to ensure the flow of resources and the easy access and exit of the personnel. This not only reduces the risk of logistical bottlenecks but also encourages the overall efficiency of the operation. Optimizing time and planning is crucial for any project, requiring proper problem definition and thorough research. With the help of innovative software, users can explore new possibilities and devise effective strategies to optimize time and planning. In this context, using drones for data collection has proven to be a game-changer. Drones can access remote areas and collect data that may not be available through traditional means. This data can be used to create detailed maps and provide valuable insights into the project's progress. Moreover, the availability of open data sources has made it easier for users to access relevant information, which can be used to enhance the project's efficiency. Combining drones and open data can lead to better-informed decisions and more accurate planning. It is worth noting that traditional cartographic maps may not always be readily available, and some may be classified or intended for military use. By collecting data from drones, users can create detailed cartographic maps that are accurate, up-to-date, and customized to their needs.
In particular, this thesis describes two GIS tools, conceited for different purposes in different situations if combined. The first is a completely offline desk tool used through the software QGIS. It allows the end user to do a visibility analysis of the field, taking as input data the chosen coordinates, the range of analysis, the Digital Terrain Model (DTM), and the established streets. The analysis outputs show all the visible zones from the chosen observation point and all the visible streets in the surroundings. This tool allows the end user to operate offline through the QGIS software and directly on the field in adverse connection conditions. The usage of this tool has a huge advantage in operation because of its versatility and scalability. It allows users to perform offline visibility analysis in minutes, saving significant planning time and facilitating swift action in the field. The second type of tool is an online mobile app that communicates with a stable position. It

allows the end user to do a visibility analysis of the field, taking as input data the chosen coordinates and the range of analysis. As the previous tool, the analysis outputs evidence of all the visible zones from the observation point and all the visible streets in the surroundings. The stable position has a Digital Terrain Model (DTM), Digital Surface Model (DSM), and streets database to detect the area. The tool offers the possibility of arriving on the territory without previous studies and preparation and rapidly responding to dangerous situations. It addresses challenges such as lack of on-site information and time constraints, offering the advantage of variable response times. Together, these complementary tools offer a comprehensive and interesting solution. These tools provide user-friendly interfaces, promote collaboration among team members, and enable real-time data transmission for dynamic scenarios, ultimately enhancing military spatial analysis capabilities.
Regarding the composition of the thesis, five chapters are present to fully immerse the lector into the various dynamics of visibility analysis. The first chapter explains the actual operational strategy of the Italian Army and evidences its gaps, describing the end-user requirements. The second chapter contains the methodology, a description of topographic and GIS environments, and the instrument used. The third chapter discusses the desk tool's realization, functionality, results, and limitations. The fourth chapter discusses the mobile tool and, as in the previous case, evidence of its realization, functionality, results, and limitations. The fifth and last chapter contains the conclusion and all the possible future development of the work done.

Chapter 1

Strategic operation: methods and tools

1.1 State of the Art

In a strategic operation, preparing and collecting information is a fundamental task. The preparation can last a few days or months, depending on the operation. Military missions assigned to the Italian Army increasingly involve deploying small and dispersed units on the ground. These units represent the tactical endpoints of the military deployment in operational theaters and serve as the first responders to potential changes that could impact the mission. Each soldier, down to the junior leadership levels, is effectively required to be a leader, capable of swiftly and autonomously making decisions that can have a significant impact, even from a strategic standpoint. To this end, they must develop a strong aptitude for information awareness, enabling them to read, contextualize, and evaluate individual events and complex phenomena within the multifaceted environment in which they operate, essentially acting as sensors themselves. These considerations apply equally to domestic deployments within national territory, where a single military personnel may perceive situational differences that could potentially impact the security of military infrastructure or ongoing operations [1]. The preparation for establishing a military strategic outpost in a specific location involves several critical phases and activities to ensure the operation's safety, effectiveness, and success.

1.1.1 Strategic Outpost

A strategic outpost can be defined as a position or location established by a military force to achieve specific objectives or to monitor and control an area of

interest. These outposts have various purposes, such as providing surveillance, gathering intelligence, supporting defensive operations, or facilitating offensive actions. The specific nature of a military strategic outpost can vary depending on the overall strategic goals of a military campaign or operation. In general, an observation strategic outpost can be permanent or temporary. The first type is the military permanent outpost characterized by previous studies of the environment and situation. His construction requires a few days with previously established resources and well-studied procedures. It can extend a few or hundreds of meters depending on the strategic necessity. A trade-off exists between having logistic supply and resources for the camp occupying a big area and the risk of being discovered or monitored from a possible threat. The decision to occupy a large or small area depends on the consciousness of the terrain and the state of the operation's alert. Personnel have to carefully consider factors such as the expected duration of the operation, the availability of resources, and the level of security required. Furthermore, the state of alert plays an essential role, as it dictates the need for adaptability and rapid response to changing circumstances.
Conversely, the temporary outpost is established in areas with limited prior information and uncertainty about the danger. His realization requires a few hours, and the level of safety and coverage has a significant degree of uncertainty. This kind of outpost requires a high level of surveillance, and the permanence time is usually one or two days at maximum. In such situations, adaptability and the ability to make rapid decisions are fundamental to ensuring the safety and effectiveness of these temporary positions. The safety of the outpost depends on momentaneous elements of coverage like vegetation or rocks. A concrete example of activity conducted to increase the situational awareness of the interest zone is the construction of a POA (Posto Osservazione e Allarme). POA can be defined as the smallest outpost of the Italian Army to collect information on the surrounding environment. The requirements of an optimal POA construction are based on various factors like the level of self-coverage, the visibility range, the access and exit points, and the radio coverage. In both scenarios, the human factor remains a critical component. Adaptation requires the ability to make rapid decisions and consider the task's priorities.

1.1.2 Strategic Operation Planning

The Italian Army follows a meticulous procedure to prepare the personnel to operate in operative theatre outside and inside Italy. Italian Army is organized into different Arms and Corps based on the everyday tasks and the nature of the study done. The Cavalry Arm plays a central role in the reconnaissance of territory and the report of information. The high availability of tactic vehicles allows them to inspect the assigned environment safely and efficiently. Moreover, those

operations follow months of training and simulation process. The training process begins with the need to replicate modern strategic scenarios as closely as possible. Simulation training of troops and units is a crucial component of operational theater studies. This type of simulation is conducted in specific simulation centers located throughout Italy, and it is complemented by practical exercises conducted on the field by various regiments. The planning phase aims to understand the field's characteristics and location. This phase begins with the search for resources and materials such as maps, satellite images, and useful software. Information is provided by the Centro Interforze di Gestione e Controllo (CIGC), SICRAL, Istituto Geografico Militare (IGM), and the intelligence service of the individual regiment. The strategic outpost positioning determination follows some guidance line. It has to be elevated to have visibility on the minor streets, control the access, and be far from big noise sources. Here are some of the key stages for the strategic outpost establishment:

- Terrain study and analysis
- Surveillance and Defense
- Operational planning
- Logistics and supplies
- Troop training

The phase of terrain study and analysis involves acquiring detailed information about the terrain, such as geographical, topographical, and climatic data, as well as information about the presence of infrastructure, water resources, and vegetation. Conversely, the terrain's characteristics, including slopes, natural cover, observation points, and obstacles, are critical to determining the ideal location. This kind of study is done by military personnel using topographic maps, satellite imagery, and data from intelligence services. The topographic map is always used by experts who analyze the elements on the map and decide on the zone to occupy for an outpost. Symbols and markers are used on maps with lucid paper to evidence the zone and elements of interest. Some accurate scale models can be realized to show the critical points of the training personnel as shown in figure 1.1. The surveillance and defense phase accomplishes the task of providing on-site information and confirming previously collected data. The awareness of what is happening in the surrounding environment is the key to identifying hidden threats and tracking enemy movements. Operational planning is fundamental to guarantee protection from enemy threats, accessibility, resource distribution, and logistics. This also allows the end user to stay in an organized environment where the synchronization of operations and communication systems is efficient. Logistics and supplies management is

assigned to specialized personnel to ensure that troops have access to necessary resources such as food, water, ammunition, and medical equipment. Finally, troop training is the base for the environment functionality. Troops must know how to behave internally to the outpost and what to do in risky situations. These phases are essential to ensure the effectiveness and safety of military operations in any operational theater.

Figure 1.1: Italian Army onfield planning phase [2]

1.2 Requirements

The use of software and technology in military operations is an essential component of the Italian Army's everyday life. In the realm of military planning, not everything always proceeds seamlessly, and the terrain can often present unforeseen challenges. Despite meticulous planning efforts, the actual field conditions may unveil numerous complexities that were not evident during the initial stages of planning. The dynamic nature of the environment, coupled with unexpected variables, can introduce a level of uncertainty that planning alone may struggle to account for. However, despite vital role of technology, some operational problems may arise, such as:

- Planning time
- Environment safeness or danger
- Communication availability

Planning a military operation requires a significant amount of time to complete. Sometimes, end-users may not have enough time to plan and conduct a field reconnaissance. The difference between gaining control of an area and gaining a competitive advantage on the field can be days or hours. In some dangerous or urgent situations, conducting a field survey or studying satellite images may not be the best choice. Even in safe situations, time is essential, as circumstances can sometimes change. In situations of danger, the commander's decisions play a significant role in the promptness of action, given the absence of information. Effective communication with remotely monitoring personnel is also essential as it allows for receiving updates and facilitates information exchange. However, in many cases, there is limited connectivity with the remote monitoring personnel. Therefore, technology plays a vital role in enhancing the responsiveness and preparedness of the military, especially in environments with limited connectivity. In addition, the safety of the environment where the military operation occurs is paramount. The use of technology can help ensure the safety of the environment. For example, drones can provide real-time video feeds of the area, which can help identify potential threats and ensure the safety of military personnel. Finally, communication availability is also critical in military operations. Effective communication with the remotely monitoring personnel can help make timely decisions. However, in some situations, there may be a lack of communication availability, making it challenging to receive updates and exchange information. Therefore, it is essential to have backup communication systems that can provide reliable communication, even when regular communication channels are unavailable. The technology helps enhance the responsiveness and preparedness of the military, ensures the environment's safety, and provides reliable communication channels even in situations where regular communication channels are unavailable.

1.2.1 User Requirements

In addressing the operational challenges faced by the Italian Army in using software and technology during military operations, it is imperative to conduct a comprehensive analysis of user requirements. By understanding the unique demands posed by military planning, the dynamic nature of environments, and the critical role of effective communication, the aim is to formulate user centric solutions that enhance the responsiveness and preparedness of the military. It is possible to provide evidence that the user may need:

- The possibility of having an automatized area analysis in planning phase
- The possibility of having rapid responses on the field
- The possibility of working offline on the field

- The possibility of having an analysis starting from a point or an established area
- The possibility of having ad analysis without any preparation done in advance

During the planning phase, users need a system that can automate area analysis, reducing the time and effort required for manual assessments. The system should provide quick and accurate insights into the operational environment, enabling end-users to receive real-time information in the field. This emphasizes the importance of timely decision-making and the need for technology that supports quick reactions to dynamic situations. Considering the limitations of connectivity in various operational environments, users require the ability to work offline on the field. This ensures that critical tasks can be performed despite reliable internet connection availability. Users also demand the flexibility to initiate analyses from a specific point of interest or an established area, making it crucial for the system to accommodate diverse operational scenarios and varying levels of preparedness in the field. The system should allow users to conduct analyses without extensive preparation in advance. This requirement acknowledges the unpredictable nature of military operations, allowing for on-the-fly assessments based on emerging circumstances. In the subsequent sections, we will explore the proposed solutions and how they align with these user-centric requirements.

1.2.2 Functional Requirements

Some functional requirements for an automated area analysis system have been identified to address these challenges and streamline the planning process:

- Ability to do an automated area analysis
- Ability to work offline on field
- Ability to work with updated cartography
- Ability to work online
- Ability to support request
- Ability to provide rapid response
- Ability to work in safe and dangerous situations
- Ability to communicate with operative center

The automated area analysis system must incorporate automated area analysis capabilities, enabling efficient and accurate planning. The system should be versatile and capable of operating on multiple platforms. It has to be accessible on personal computers for detailed planning and on smartphones to facilitate rapid responses and field analyses. Moreover, the system's desk tool component has to be able to work offline. The system must seamlessly integrate with digital cartography, allowing for efficient geospatial information processing. Furthermore, it should be adaptable to work with diverse datasets acquired through open data systems or directly from the field, including through the use of drones. This ensures that military personnel can conduct critical analyses even in environments with limited or no internet connectivity, providing resilience in various operational scenarios. In emergency and unplanned situations, a dedicated mobile tool is essential. The tool has to enable quick responses, and analysis requests directly from smartphones. Furthermore, both the desk and mobile tools should feature a user-friendly interface. This ensures accessibility for military personnel with varying technical expertise, promoting ease of use in dynamic and high-pressure situations. The system has to facilitate real-time communication between field personnel and command centers. This includes efficiently transmitting analysis results, updates, and information exchange, enhancing overall coordination during military operations. The mobile tool has to support on-demand analysis requests, allowing users to quickly initiate analyses in response to emerging circumstances. This capability enhances the system's responsiveness in unpredictable situations. The proposed system provides the Italian Army with a robust technological solution that enhances responsiveness and adaptability in diverse operational scenarios.

1.2.3 Technical Requirements

It is essential to define some technical requirements to ensure the proper functioning and compatibility of the automated area analysis system. Technically, the system requires:

- Compatibility with standard hardware specifications
- Android operative system
- GIS software
- Strong database
- User authentication

The system should be designed to work with commonly used hardware configurations to reduce potential compatibility issues and ensure accessibility. Compatibility

with the Android operating system is essential for the mobile tool as it enhances flexibility and supports a wide variety of devices, making it efficient for deployment among military personnel. Integration with Geographic Information System (GIS) software is crucial for spatial analysis and planning. The system should have GIS capabilities to provide detailed insights into geographical data, facilitating informed decision making. A well organized and robust database is essential for storing and managing diverse geospatial data, including digital terrain models and road information. Secure user authentication mechanisms must be implemented to control system access, ensuring only authorized personnel can use its features. This is crucial in maintaining the confidentiality and integrity of sensitive military information. Efficient internet data transmission capabilities are necessary for real time updates in way to maintain situational awareness and facilitate timely decision-making.

Chapter 2

Digital Maps and Open Data

2.1 Digital Map and Elevation Model

2.1.1 Traditional Map

A map can be defined as a drawing of the earth's surface, or part of that surface, showing the shape and position of different countries, political borders, natural features such as rivers and mountains, and artificial features such as roads and buildings [3]. A large-scale traditional technical map comprises a territory representation divided into sheets and completed with appropriate frames and parameters. The map is created in a system of flat cartesian coordinates whose points correspond biunivocally, based on precise geometric and mathematical relationships, with the physical surface of the represented territory. Additionally, it should be noted that the representation of points can be divided into two distinct categories of information: planimetry, involving horizontal coordinates (x, y), and altimetry, focusing solely on the vertical component. The planimetry is formed by the projection onto the drawing plane of the natural and artificial details of the terrain. The altimetry or vertical component it is composed of contour lines and known points in the vertical component. The planimetry is always present, while the altimetry may be absent, and in this case, the map is only planimetric. Managing the vertical component presents various approaches in two dimensional maps. The first approach includes the use of quoted points, providing a specific altimetric reference for each map point. Contour lines represent another option, connecting points at the same altitude and offering a visual indication of terrain elevation changes. Additionally, in more complex situations, shading could be employed as a method that uses different tones to represent altimetric variations, providing an immediate visual perspective on the topography. Other characteristic elements that define the drawing as a traditional map include the scale ratio of *1:n*, where n is the number of times the topographic distance between two points is reduced on

the map. The legend also provides the key to interpreting the map based on types of lines, hatching, symbols, conventional signs, etc. Finally, the metric content must comply with tolerances, establishing the maximum deviations between the distances and elevations derived from the map and those in reality.

2.1.2 Digital Map

Digital cartography provides qualitative and metric information inherent in traditional cartography under two aspects: in the form of numerical data (coordinates describing the geometry of mapped objects and codes indicating their types), stored on magnetic media processed by an electronic computer, and in the form of visualizations on a video-graphic display or paper through a plotter, similar in appearance to traditional cartography. Therefore, digital cartography constitutes a mirror image of traditional cartography, as the latter is a cartographic product in the form of a drawing that implicitly contains the same data in the form of coordinates. While traditional cartography comprises a drawing that implicitly contains the same data as coordinates, numeric cartography consists of a coordinate archive that implicitly contains its visualization as a drawing. Digital cartography also transforms cartographic data into information based on the logical processes of a human operator using one of its possible visualizations and based on computations programmed and performed by the electronic computer. Furthermore, complete uniqueness to the metric content of cartography is provided by eliminating both the subjective elements that affect the measurement operations in traditional cartography, where one transition from drawing to coordinates, and the consequences of the deformability and deterioration of paper supports. Uniqueness is also total from the qualitative content perspective, as the interpretation of the drawing is replaced by reading the coding. Finally, digital cartography allows for the expansion of the typology of cartography, introducing other types of cartography based on an increase in altimetric information. This is made possible because, formally, there is no longer a difference between planimetric and altimetric information.
A digital map contains different data types classified into object, entity, and geometric element. The object is any natural or artificial element that is not more divisible, like a lake, a house, or a wall. The entity regards the cartographic representation of the object and can be defined as polygons (closed polylines), polylines, or points with associated codes. It can be formed from one or more geometric elements. The geometric element is constituted by a polyline with an associated code. Furtermore, a digital map can be planimentric (2D), plano altimetrical(2.5D) and tridimetional (3D). The planimetric map describes only the elements that characterize the planimetry through the two coordinates (E, N) and the relative coding. Similar to traditional cartography, in the plano-altimetrical map, the planimetry is described separately from elevation; the first through two coordinates

(E, N), the second through three coordinates *(E, N, h)*. The tridimensional digital map describes each point by three coordinates *(E, N, h)*. The classic description of elevation consists of contour lines and spot heights.

2.1.3 Nominal Scale

One of the most innovative aspects in the context of cartographic tradition is that, for digital mapping, the concept of scale ratio seems to be surpassed. This is because, aside from the deformation factor introduced by the adopted map projection, the Pythagorean theorem applied to the planimetric coordinates of two points directly provides their topographic distance; in other words, the terrain is stored in the form of absolute coordinates, and therefore is always and consistently at a *1:1* scale. It might appear improper, then, to speak of a scale for digital maps. However, it is believed that this reference should not be abandoned, especially considering that digital cartography data can be visualized via plotters or on graphic screens with exaggerated zoom-ins and zoom-outs. This implies that digital cartography adhering to the same cartographic principles as a traditional map at a *1:2000* scale could be easily displayed or scaled up, for example, to a *1:500* or even *1:200* scale, potentially leading the user to believe they possess a precision that is not inherently present. It is crucial for users to understand that digital cartography has a very specific metric precision dictated by classical photogrammetry standards. Therefore, expecting to acquire more information by operating on visualizations at scales greater than the nominal scale is akin to the error made by a traditional cartography user who believes they can increase its informational and metric content by enlarging a photographic print. It is therefore agreed that the concept of scale ratio should be maintained for digital cartography, and the scale ratio for digital cartography should be understood as the maximum ratio at which a digital map can be reproduced using a plotter, ensuring it meets the qualitative and metric requirements of a traditional map at the same scale. For digital cartography as well, the scale ratio thus defines the degree of metric precision and qualitative content. For formal correctness, it would be appropriate to use the term nominal scale for digital cartography, intending this to be the scale ratio that a traditional map with corresponding metric precision and qualitative content would have. As a partial correction to the considerations regarding enlargements of digital cartography at scale ratios higher than nominal, it is worth noting that visualization on a graphic screen allows, or rather encourages, the representation of digital cartography at scales higher than nominal. Therefore, while any enlargement beyond the nominal scale of the map should be considered at the user's risk, it would be incorrect to consider the display on a graphic screen at a scale higher than nominal in the same light as the criticized photographic enlargement of a traditional map. The ease of automatically producing visualizations of digital cartography on a graphic screen

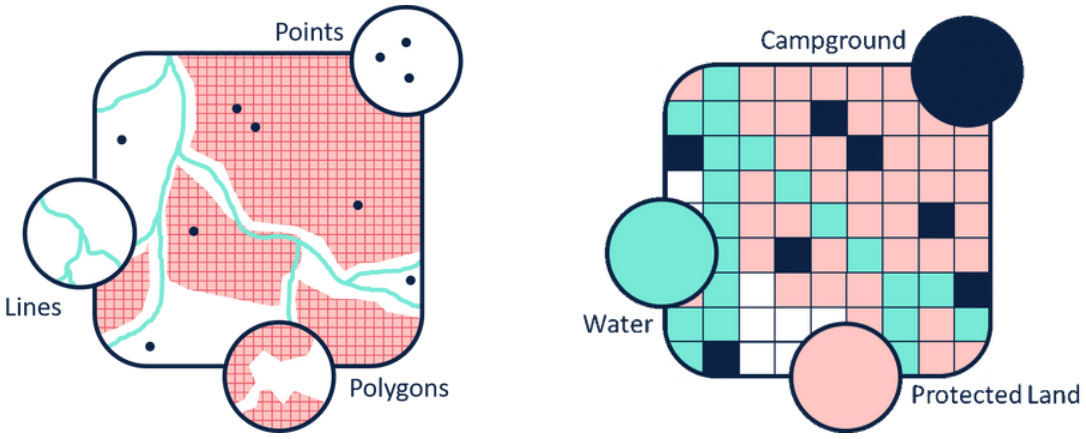
or via a plotter at scale ratios higher than nominal should be seen as an additional opportunity offered by digital cartography to read not more than what it actually contains but to read it better.

2.1.4 Raster and Vector

The digital map, in addition to the set of numerical data, also includes two forms of graphical representation:

- Vector
- Raster

Vector is a type of representation through which it is possible to describe an object's geometry using only points, lines, and polygons. A clear example of this feature is evidenced in figure 2.1a. Vector data provides two pieces of information: the geometry (graphical component) and the effective data (x,y,z). The graphical component describes the object's shape and appearance, while the effective data provides information about the object's location, size, and orientation. The advantage of using vector data is the possibility of describing only the elements of interest and creating a realistic object shape. In the database, it is possible to include several fields (columns) that describe different properties of the object.



(a) Vector Data Concept **(b)** Raster Data Concept

Figure 2.1: Vector and Raster Differences [4]

Raster is defined as the representation of an object as regular elements (cell/pixel), where each cell or pixel carries a unique value. The cell size in defines the spatial resolution or granularity of the data. It represents the physical size of each cell or

pixel in the raster grid. A smaller cell size provides higher spatial resolution, allowing for a more detailed representation of the geographic features. A clear example of this feature is evidenced in figure 2.1b. For each cell it is applied a digital number representing the value associated with the geographic attribute being measured or observed. These values could represent the temperature at a specific location, the elevation of the terrain, or the intensity of a particular spectral band in a satellite image. For example, in a satellite image, each pixel might have a digital number corresponds to the amount of reflected sunlight in a specific wavelength range. The digital number is often determined by the sensor's ability to capture and quantify electromagnetic radiation. The intensity of the signal detected by the sensor is converted into a digital value, and this value is assigned to the corresponding pixel in the raster dataset. The digital number assigned to each pixel serves as a quantitative representation of the observed phenomenon. The process involves calibrating the sensor data to convert raw radiometric values into standardized numbers. This calibration ensures that the digital number values accurately reflect the real-world characteristics captured by the sensor. The combination of spatial resolution (determined by cell size) and radiometric resolution (determined by bit depth or levels of sensitivity) in raster data allows for the creation of detailed and accurate representations of geographic features, making it a fundamental tool in fields such as remote sensing and geographic information systems (GIS). The advantage of raster data is the complete object description without focusing only on the edges. The coordinates of the pixels are regular, and it is not necessary to store them for each pixel, but only the position of the single pixel. Raster data also allows the application of mathematical functions because it is a matrix.

2.1.5 Elevation Model

In traditional cartography, it is necessary to use interpolation procedures that provide the data height depending on the elevation information represented by elevation points and the contour lines to identify the part of a generic point. The digital map overpasses the limit of traditional cartography, proposing two ways to describe the elevation. The first one regards directly the elevation in the coordinates. In contrast, the second one regards creating a specific Digital Model that describes the elevation of the terrain or surface point by point. Differently from traditional maps, the tridimensional data that describes the elevation are classified into the list of elevation points, contour lines, height of points that describe the entities, and height description of volumetric units. Digital models can be classified into the Digital Elevation Model (DEM)/ Digital Terrain Model (DTM) and the Digital Surface Model (DSM). A DEM, or equally a DTM, can be defined as a cartographic model that includes a logical mathematical description of the terrain objects in

digital form and contains data about their characteristics. It is created in the established map projections, scale, coordinate systems, and elevation systems, considering the principles of cartographic generalization and establishing necessary topological relationships between objects. [5] The elements that compose a DTM can be seen in 2.2:

- Points that can be random or regular inside the model.
- Breaklines(in red), which are characteristic lines of the terrain, buildings, or surface and placed where there are strong values of slope variation.
- Chorographic elements are specific points where it is fundamental to describe the elevation, like the top of a mountain.
- Dead zone(in yellow), described as a region where I do not have any points or information.
- Limits of the model(in blue), used to delimit the model borders.

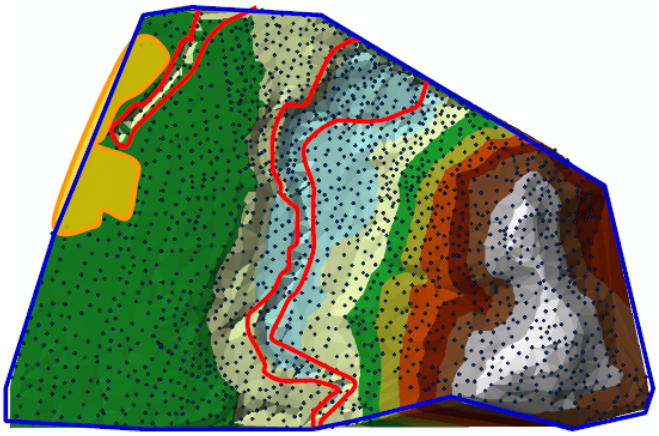


Figure 2.2: DTM Elements

On the other hand, a DSM can be defined as a topographic digital model that describes the relief and the situation on the surface. It consists of a digital terrain model representing the Earth's surface, including all objects. [6] The main difference between DSM and DTM lies in what is included in the model. A DSM includes all features on the Earth's surface, both natural and artificial, while a DTM focuses solely on the natural topography, excluding above-ground objects. Depending on the specific needs of a project or application, one might choose to work with either a DSM or a DTM. An example of comparison between DEM and DSM is evidenced in figure 2.3.

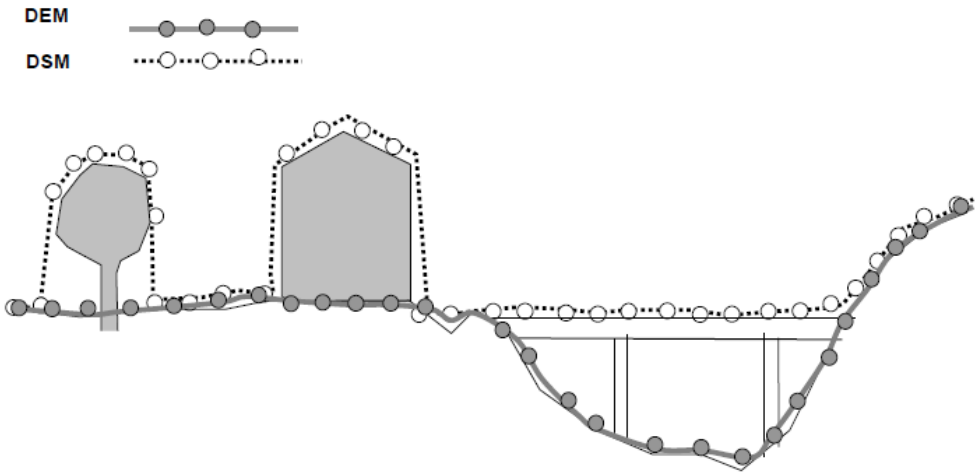


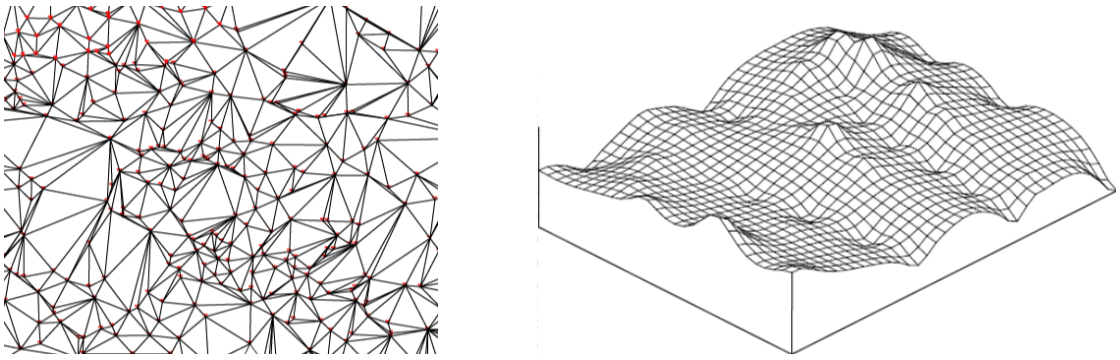
Figure 2.3: DEM and DSM Comparison

On the other hand, when the model contains a large number of points, it is not imperative to provide a meticulous and comprehensive description of the surface. Detailed elements like breaklines and chorographic features become non mandatory under such circumstances. Various Dense Digital Elevation Models, known as DDEM or Dense Digital Terrain Models (DDTM), and Dense Digital Surface Models (DDSM) should be consistently incorporated. It is crucial to always include a Dead Zone, delineating the limits of the specified area with those specific models. In particular, the DDTM is a more detailed version of the DTM, characterized by a higher density of data. This model is particularly suitable for capturing intricate details of the terrain, such as small elevations, valleys, or ridges. The DDSM is a more detailed version of the DSM, capturing a wide range of elements present on the surface with a higher density of data points. This model is useful in applications that require an extremely detailed representation of the visible surface.

2.1.6 Point Distribution Models

The distribution of points in a Digital Elevation Model exhibits an irregular pattern, and different methods are employed to represent this distribution. The Point Cloud method represents a set of three-dimensional coordinates (Easting, Northing, Elevation) distributed in an irregular pattern. This representation provides precise details about the spatial positions of points but does not inherently offer information about the surface between them. The Triangulated Irregular Network (TIN) is a method that connects points in the point cloud using triangles through triangulation. This method can be observed in figure 2.4a. Each triangle in a TIN represents a planar surface, and the network of triangles allows for a more

accurate representation of the terrain or objects. However, TIN is not univocal, meaning there can be different configurations of triangles connecting the same points. Clearly describing the arrangement of triangles is essential to ensure the accurate interpretation of the data. Moreover, it is possible to consider a regular distribution or grid, where the points have a consistent spatial resolution. This method can be observed in figure 2.4b. The resolution must be adequately reduced to enable linear interpolation but not excessively small to minimize file size and facilitate straightforward data processing. The resolution of this grid must be carefully chosen. It should be reduced enough to facilitate linear interpolation for a smooth representation of the data, ensuring that the transitions between cells are visually seamless. However, it should not be excessively small to strike a balance between detailed representation and practical considerations such as file size. An optimal resolution allows for efficient data storage and processing, making it easier to manage and analyze spatial information effectively. The grid model can be defined in two distinct ways. One approach involves cartographic coordinates, characterized by meter level precision, making them suitable for small portions of territory. On the other hand, geographic coordinates utilize degrees and are more applicable to larger expanses of land. It's important to note that a regular grid in cartographic coordinates does not align with a regular model in geographic coordinates, and vice versa. As a result, there is a need to establish a new interpolation method to bridge the gap between these coordinate systems.



(a) TIN Representation Concept **(b)** Grid Representation Concept

Figure 2.4: TIN and Grid Differences

2.2 Cartography Availability

In this context work, it is essential to explore the variety of cartographic resources available to the user. This section evidences existing cartography, drawing from

sources defined by the Italian law 2/2/1960 n° 68. That law defines Italian cartographic organ authorities the following ones:

1. Istituto Geografico Militare (IGM)
2. Istituto Idrografico della Marina (IIM)
3. Photo-cartographic section of the Air Force (CIGA)
4. Land Register Agency (Catasto)
5. Geological Service
6. Region Authority

The role of the Istituto Geografico Militare (IGM) in the history of military and Italian cartography as a continuous and actual provider of cartography data is worth mentioning. Most cartographic data are also provided by Regional Geoportals and portals, products provided by the Ministry of Environment and Territory, open source cartography projects like OpenStreetMap (OSM) and Remote Sensing Services.

2.2.1 IGM

IGM plays a fundamental role in the production of Italian cartography. Founded with the primary objective of providing cartographic support for military needs, the IGM has evolved over the years to become the authority responsible for producing and updating official maps of the Italian territory. Its expertise extends to a wide range of cartographic products, including detailed topographic sheets, thematic maps, and nationally relevant geographic data. Thanks to advanced technologies and innovative surveying methods, the IGM continues to play a key role in providing a reliable and accurate cartographic foundation, essential for military, civilian, and territorial planning activities in Italy. The Istituto Geografico Militare (IGM) produces a broad range of topographic maps. It is essential to mention some of these topographic maps produced today that cover the whole of Italy. In the current edition of topographic maps, the national territory is divided into 636 map elements at the scale *1:50,000*, known as sheets. Each sheet has dimensions of 20' in longitude and 12' in latitude. Subsequently, each sheet is further divided into four equal parts at the scale *1:25,000*, known as Sections. These sections are 10' in longitude and 6' in latitude. Roman numerals identify each section. An excerpt of this map at the scale *1:50,000* can be seen in the figure 2.5.

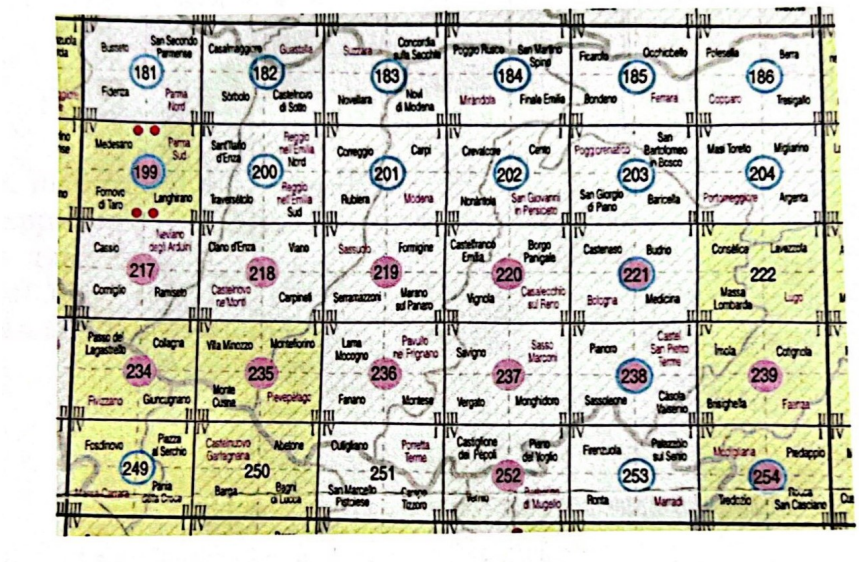


Figure 2.5: Excerpt from the union table of the map at 50000 as it appears in the IGM catalogue

It is also worth mentioning some of the Topographic Map Series of Italy such as series 25 and series 25DB. The Series 25 Topographic Map of Italy represents cartography at the scale *1:25,000*. It is currently no longer in production at the Institute, as the 25DB series continues and replaces it. This cartography consists of 2,298 elements, of which 840 have been completed and are referred to as sections. Each section has dimensions of 6' in latitude and 10' in longitude. The sections are processed using aerophotogrammetric surveys, both numerical and analogue, and subsequently drawn using automatic or manual methods. They are framed in the conformal representation Universal Transverse Mercator (UTM), and the geodetic reference system is based on the international ellipsoid with European mean orientation (ED 1950). This cartography presents the orography with contour lines at an equidistance of 25 metres and indicates state borders as well as regional, provincial and municipal administrative boundaries. It is printed in four colours. An example of this topographic map can be seen in the figure 2.6a. The Series 25 DB Topographic Map of Italy continues the previous Series 25, identifying cartography at the scale *1:25,000*, currently in production at the institute. This map consists of 2,298 elements called sections, each of which is 6' in latitude and 10' in longitude. The Series 25 DB replaces and continues the previous Series 25. The sections are obtained by numerical stereorestitution or are derived from the technical cartography of the regional kilometric grid in Mercator's Universal Transverse Conformal Projection. The geodetic reference system is ETRS89, which uses the GRS80 ellipsoid. The information content includes human works,

hydrography, vegetation and orography. An example of this topographic map can be seen in the figure 2.6b.

(a) Series 25 Example **(b)** Series 25DB Example

Figure 2.6: Topographic Map of Italy: Series 25 and 25DB

2.2.2 Regional Geoportals

Today, cartography is available on different platforms or websites, allowing users to work with numerous and high quality data. One of the primary services providing that information is regional geoportals. Regional portals are websites or dedicated applications offering a wide range of geographic information, including maps, topographic data, and thematic layers. Regional portals serve as centralized digital platforms that provide access and utilization of geographic data at the regional level. They provide centralized access to detailed geographic data, encompassing maps, satellite imagery, topographic information, and other region-specific thematic data layers. Additionally, they offer interactive visualization tools, enabling users to dynamically explore geographic information such as zooming into specific areas or selecting specific data layers. They may also allow the download of geographic data for later use and facilitate information sharing across various platforms, contributing to the dissemination of geographic knowledge. An example of an Italian geoportal is the Geoportale della Regione Piemonte. This portal provides access to a wide range of region specific geographic data, offering interactive tools and services for exploring and understanding the Piedmont territory. Users can access thematic maps, topographic information, environmental data, and more through a user friendly platform. It is necessary to mention some topographical maps issued by

the Italian regions called Carta Tecnica Regionale (CTR). These maps, depict a detailed survey of the Italian territory, providing precise information on topography, administrative boundaries, watercourses, and other relevant elements. The *1:5.000* scale strikes an ideal balance between detail and geographic coverage, making these maps useful for a variety of applications, from hiking activities to military operations and territorial planning. An example of this map is evidenced in figure 2.7.

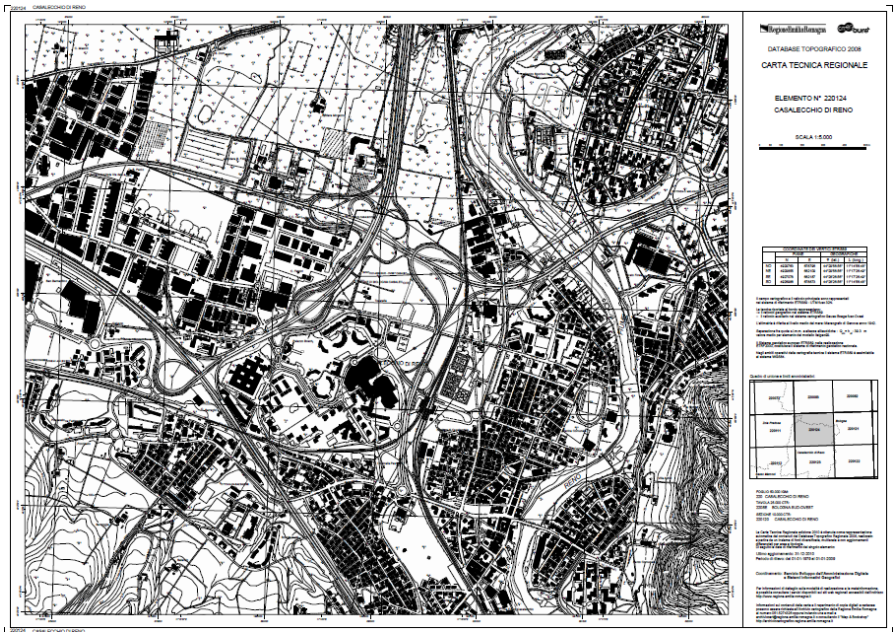


Figure 2.7: Carta Tecnica Regionale (CTR) Example [7]

2.2.3 Ministry of the Environment and Territory Products

On the other hand, the products of the Ministry of the Environment and Territory in Italy provide detailed geographic data and cartography, along with environmental information, ecosystem monitoring, and regulations. They also encompass tools for territorial planning, online services, and digital platforms for accessing and sharing information. These products contribute to the formulation of environmental policies, the implementation of research and monitoring programs, as well as the promotion of public participation in sustainable land management. In general, they reflect the Ministry's commitment to environmental sustainability and the protection of the national territory. An example of products provided by the Ministry of the Environment and Territory in Italy could include the Sistema Informativo Nazionale per la Gestione e la Conservazione del Territorio e dell'Ambiente (SINT). This system offers a broad range of data and services for national level land management

and environmental conservation.

2.2.4 Open Cartography

One other service providing geographic information is open cartography, exemplified by OpenStreetMap (OSM). OSM operates as a collaborative mapping platform shaped by a global community of users who voluntarily contribute to the collection, updating, and sharing of geospatial data worldwide. This open participation model results in a vast and diverse collection of geographic information, encompassing details such as roads, buildings, green areas, and much more. The platform's flexibility is reflected in a range of visualization and editing tools, allowing users to explore and update data directly on the web platform. OSM data is widely used in navigation applications, geolocated services, and territorial analyses, powering a variety of contexts and sectors. Its open license promotes the freedom to use the data, requiring only proper attribution and the sharing of any derivative works under the same license.

2.2.5 Remote Sensing Services

There are also some services which provide detailed and timely data on a wide range of environmental parameters, contributing to understanding changes, predicting phenomena, and supporting informed decision-making across various sectors. Copernicus and other remote sensing services play a pivotal role in monitoring and analyzing the Earth's environment from a satellite perspective. Copernicus, in particular, stands out with its constellation of Sentinel satellites, covering diverse disciplines such as atmospheric observation, land monitoring, marine monitoring, and more. These data, made accessible to the public, address critical challenges like climate change, natural resource management, and response to catastrophic events. Similarly, other remote sensing services offer specialized solutions. For instance, Landsat, managed by NASA and the USGS, provides long term images of the Earth's surface, valuable for monitoring changes in environments such as forests, agriculture, and urbanization. European Sentinel satellites and remote sensing services from various nations contribute to a global and collaborative framework for environmental monitoring.

2.3 GIS Environment

Over time, and especially with the rapid technological progress, digital cartography has been required to support applications typical of a computer system, namely the ability to manage the generated information. This has led to the creation of the Geographic Information System (GIS). An Information System (IS) can

be defined as a set of tools and applications able to manage, collect, store, and distribute information to territorial entities. A Geographic Information System (GIS) is an Information System based on technologies able to acquire, update, store, model, manipulate, extract, analyze, and visualize data that are spatially located (geo-referenced). A Geographic Information System is composed of:

- Cartographic support/map Database
- GIS management software
- Hardware tools for input/output
- Organization framework

GIS is not only the usage of information but a process composed of several steps to reach the definition of the GIS model. GIS model creation requires some steps to satisfy. The first one is the definition of the problem to satisfy through GIS. Consequently, there is the question of defining the GIS criteria with all the objects needed in my model. The third step is to import or build a dataset that describes my objects. The fourth step is the main one and represents the data analysis to arrive at one output that can be a plot or a tridimensional model. The last step is to decide how to complete the definition of my model. In this way, GIS encompasses a structured workflow to address and solve spatial problems effectively.

2.3.1 GIS Data

A georeferenced data describes a real object in terms of position, attributes, and relationships. This type of data is critical in GIS as it allows for creating comprehensive and informative geographic analyses. There are three different types of GIS data:

1. Geometric Data
2. Attributes
3. Metadata

Geometric Data are used to describe the shape of an object and can be classified into raster and vector data. In a GIS environment like QGIS, raster and vector data types are used to create comprehensive and informative geographic analyses, making it a versatile tool for various spatial projects. Attributes or descriptive data provide additional information about the geographic elements of geospatial data, such as names, values, and dates. Attributes are essential in analyzing geospatial data as they allow for identifying patterns and relationships, enabling users to

make informed decisions. Metadata is the essential information that describes, documents, and provides context for other geospatial data. It includes information such as data source, projection, accuracy, and creation date.

2.4 Open Data

Used data provenience is fundamental to having the right resources to work cleanly. Open data refers to readily available data for use, reuse, and redistribution by anyone, subject only to the requirement of attribution and sharing. Access to open data is provided under specific data license conditions and can be modified and shared according to the license terms. There are three primary methods for sharing open data, including:

- Free download of the data: This method involves downloading data on a computer as raster, vector, or tabular data. This approach allows for offline work and centralizes all data in one location, but it may create memory size issues. Raster data is used to represent images, while vector data is used to represent objects with geometrical attributes. Tabular data is used to represent information in a table format.

- Visualize the data on the web: This method involves visualizing data on a web platform. It has the benefit of having updated data without downloading, but it requires a good internet connection. This approach is typically utilized when sharing data with a broad audience, such as the public.

- Use a web service: This method involves using data-based web services on a server that can be visualized and used in the software with a stable internet connection. Three types of services are provided: WMS (Web Map Service), WFS (Web Feature Service), and WCS (Web Coverage Service).

Geospatial data is a crucial domain that values openness, collaboration, and accessibility. The fundamental principles of the open data ecosystem are centered around open platforms, protocols, and licenses, which form the foundational elements of this field. To understand the seamless flow of geographic information across various applications and users, it is essential to have a comprehensive understanding of these components.

2.4.1 Open Platforms

In computing, an open platform describes a software system that is based on open standards, such as published and fully documented external Application

Programming Interfaces (API) that allow the use of the software to function in other ways than the original programmer intended, without requiring modification of the source code. [8] Open-source or collaborative platforms supporting the management, analysis, and sharing of geospatial data are, for instance, open-source GIS platforms like QGIS or GeoServer, allowing users to work with geographic data without licensing restrictions. Open platforms are designed with an open and flexible architecture, allowing the integration of various technologies, data, and functionalities. The main goal of an open platform GIS is to promote the sharing, accessibility, and interoperability of geospatial information and guarantee a collaborative and flexible approach in the use of GIS technologies. Concentrating mainly on QGIS as one of the most famous open platforms is essential. QGIS is a user-friendly Open Source Geographic Information System (GIS) licensed under the GNU (General Public License). QGIS is an official project of the Open Source Geospatial Foundation (OSGeo). It runs on Linux, Unix, Mac OSX, Windows, and Android and supports numerous vector, raster, and database formats and functionalities. It provides a continuously increasing number of capabilities provided by core functions and plugins. It allows visualizing, managing, editing, analyzing data, and composing printable maps.[9]

2.4.2 Open Protocols

Open protocols are communication protocols that are accessible and usable by anyone. In the context of Geographic Information Systems (GIS), these protocols are mainly developed and maintained by the Open Geospatial Consortium (OGC). This international consortium creates open standards for geospatial and location-based services. Standardized open protocols promote interoperability among different geospatial systems, allowing them to work together seamlessly. Some OGC Services evidenced in figure 2.8are:

- Web Map Service (WMS) Web Feature Service (WFS)
- Web Coverage Service (WCS)
- Sensor Web Enablement (SWE)
- Catalog Services - Web (CS-W)
- Geospatial Semantic Web (GSW)
- Semantic Sensor Web (SSW)

As discussed in section 2.4, it is crucial to describe the open protocols used in the field of geospatial data services, such as the Web Map Service (WMS), Web Feature Service (WFS), and Web Coverage Service (WCS). The Web Map Service

provides map images that can be overlaid with different data layers. It allows clients to request and receive static map images, usually raster images, that can be displayed in a mapping application. The Web Feature Service provides vector data that can be edited and queried. This protocol enables clients to request and receive geographic features like points, lines, and polygons as vector data. The Web Coverage Service provides data in a gridded format. It allows clients to request and receive coverage data representing information covering a geographic area. This service is typically used for accessing and exchanging raster data, like satellite imagery or other gridded datasets.

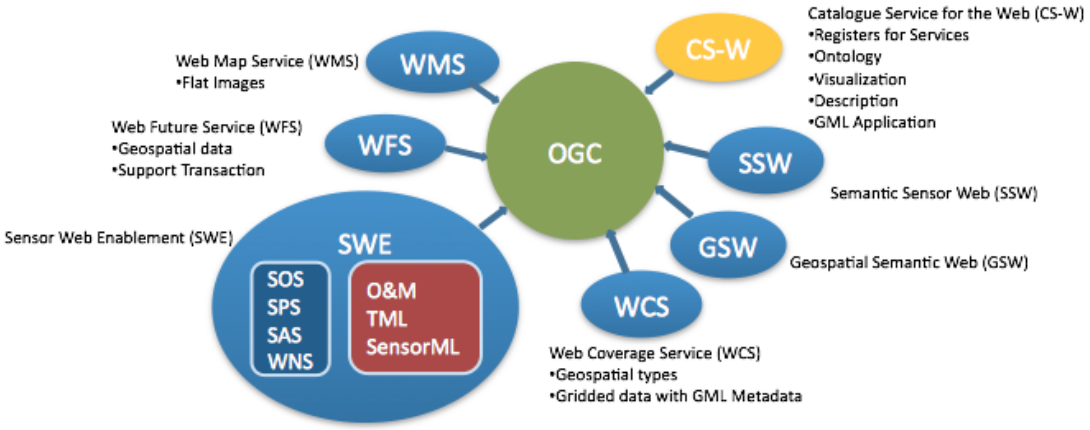


Figure 2.8: OGC Standards [10]

2.4.3 Open Licences

Open licenses permit the use, modification, and sharing of works or resources. Such licenses promote the freedom to use, adapt and distribute resources, encouraging accessibility and collaboration. In the context of Geographical Information Systems (GIS), these agreements are commonly applied to software, geospatial data, and maps. For example, QGIS adopts open-source licenses such as the GNU, while geospatial data may be shared with licenses like the Creative Commons Attribution (CC BY). The primary objective is facilitating the free exchange of resources promoting sharing and cooperation within the geospatial community. Figure 2.9 shows different types of software licences and their features. It is important to mention the GNU used in QGIS. It can be defined as a series of widely used free software licenses that guarantee end users the four freedoms to run, study, share, and modify the software. The license grants the recipients of a computer program the rights of the Free Software Definition. These GNU series are all copyleft licenses, which means that any derivative work must be distributed under the same or equivalent license terms. [11]

	APACHE	BSD	MIT	GPL Free as in Freedom	LGPL Free as in Freedom	AGPL Free as in Freedom
Type	Permissive	Permissive	Permissive	Copyleft	Copyleft	Copyleft
Provides copyright protection	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Can be used in commercial applications	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE	✓ TRUE
Provides an explicit patent license	✓ TRUE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE	✗ FALSE
Can be used in proprietary (closed source) projects	✓ TRUE	✓ TRUE	✓ TRUE	✗ FALSE	✗ FALSE partially	✗ FALSE for web
Popular open-source and free projects	Kubernetes Swift Firebase	Django React Flutter	Angular.js JQuery, .NET Core Laravel	Joomla Notepad++ MySQL	Qt SharpDevelop	SugarCRM Launchpad

Figure 2.9: Software Licences [12]

Chapter 3

GIS Desk Tool

The GIS Desk Tool is a highly versatile tool that is designed to cater to a wide range of needs and to meet specific expressed requirements. In particular, it has been developed to operate entirely offline, both in the field and remotely, providing prompt responses. The tool requires preloading the Digital Terrain Model (DTM) or Digital Surface Model (DSM) for the chosen area and the file containing all the roads surrounding the area. Once these data are stored in a computer database, the end user can go to the area of interest and obtain the offline visibility analysis in just a few minutes. There is also the option to download data on the spot, run the tool, and transmit the results to the field through simple data communication. This tool is a versatile solution capable of operating seamlessly in both on-site and remote settings, delivering swift responses. Primarily, it allows significant time savings in the planning phase and enables swift action in the field. It provides the option to conduct visibility analysis from a selected point or within a defined radius. The decision to define a radius from the observation point addresses the practical consideration that, in the field, users might not always be aware of the specific polygon where they intend to establish an outpost. By allowing users to input a radius value, which represents the radius of a circle centered at the observation point, the tool accommodates real-world scenarios where precise polygon definitions may not be immediately evident on-site. This feature enhances the tool's usability, enabling users to quickly and effectively perform visibility analysis without the need for detailed polygon knowledge on the field. The analysis from an area will be the combined results of the visibility of all the surroundings and streets. An additional advantage of the tool is the focus of results on roads, as it highlights the road segments visible and invisible from the observation point. Expanding on the GIS desk tool, it offers a user-friendly interface, making it accessible to professionals with varying technical expertise. It also facilitates collaboration by allowing users to share analysis results with team members, promoting efficient decision-making processes in field and remote environments.

3.1 Functionalities

3.1.1 Flow Chart

Figure 3.1 presents a detailed overview of the tool's flow chart, primarily divided between input parameters and algorithms. The first green box highlights the eight possible input parameters, while the other light blue boxes represent intermediate algorithms. The light green boxes represent the algorithms, and the orange boxes represent optional points. The ouputs are represented by the green box on the right. Within the first box, six mandatory parameters and two optional parameters are observed. Specifically, these parameters are:

- Observation Point
- Observation Radius (optional)
- Analysis Radius 1
- Analysis Radius 2 (optional)
- Elevation Model (DTM or DSM)
- Streets
- Observer Height
- Target Height

The Elevation Model serves as the core input for all algorithms. The Observation Point represents the coordinates of the location for visibility analysis, manually inputted and processed through the Point Creation algorithm to become an actual Observation Viewpoint. The input parameters also include Streets, a vector layer representing the area's relevant roads. The first phase is to create the points in those streets through the Streets Point algorithm and have a visible output. Together with the Elevation Model, Analysis Radius 1 or 2, Target Height, and Observer Height, it feeds into the Streets Viewpoint algorithm. The Observation Radius is an optional parameter, expressed in meters, subject to algorithms that create multiple observation points around the Observation Point, facilitating visibility analysis in a specific area. This parameter is optional as the analysis is conducted in this manner only if the field is filled. Regardless of whether a single point or multiple points are used as the basis for the analysis, they serve as input for the Observation Viewpoint algorithm. This algorithm takes as input the Elevation Model, Analysis Radius 1, Target Height, and Observer Height. Analysis Radius 1 represents the radius in meters that defines the extent of the desired area analysis. The output of the Observation Viewpoint algorithm becomes the input for the Viewshield Analysis

algorithm. The Viewshield Analysis, taking this result along with the Elevation Model, performs visibility analysis across the desired area. Finally, the Streets Viewpoint, combined with the Observation Viewpoint, serves as input for the Intervisibility Network 1 algorithm. Optionally, the Analysis Radius 2 parameter can also be inserted, allowing for a second visibility analysis for comparison. If filled, it allows the creation of Streets Viewpoint 2 and Observation Viewpoint 2 to perform Intervisibility Network 2. The Intervisibility Network algorithm has the capability to highlight the roads that are visible and not visible within the chosen radius of analysis.

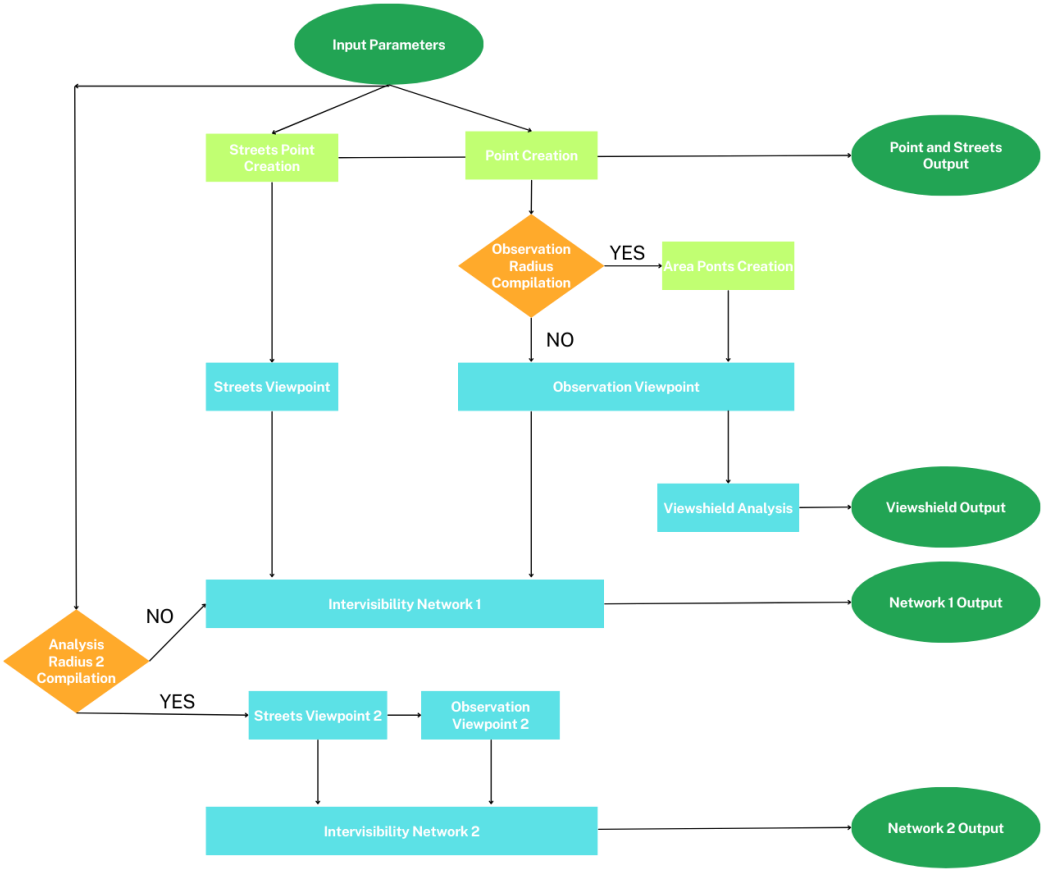


Figure 3.1: GIS Desk Tool Flow Chart

3.1.2 Tool Functionalities

Depending on the user's specific requirements, its various functionalities can be configured differently to achieve different results. The table 3.1 shows the input parameters of the four possible configurations. In the first configuration, the tool

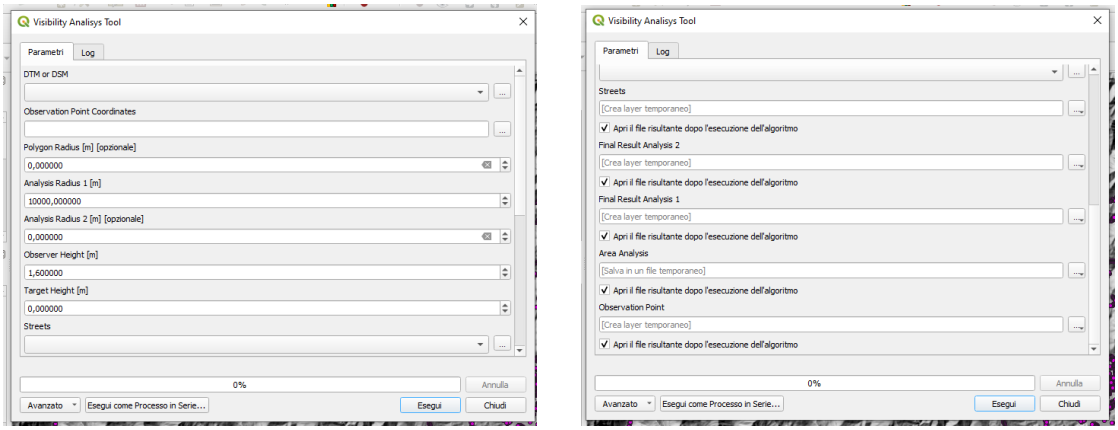
takes the input coordinates of an observation point and uses a series of advanced algorithms to perform a detailed visual analysis. This analysis shows the visible and non-visible areas from the observation point within a specified radius. The tool also creates an Intervisibility Network between the observation point and the roads. The visual output includes a detailed map highlighting the visible and non-visible roads in the observed area and the roads provided as input and the observation point. In the second configuration, the tool allows the user to set a radius for the chosen polygon as the observation area. The tool generates various points based on the input coordinates and radius and performs a highly detailed visibility analysis on each point. The cumulative observation of all points is considered, and the tool produces an expected result that provides a comprehensive view of the observed area as in the first configuration. The third configuration is designed to provide more flexibility to the user. It allows the user to set an observation point by entering its coordinates and the first and second radius to visualize the area. Based on these inputs, the tool performs two analyses with different visualized areas. The visive output is the same as the first configuration but with two overlapped analyses colored differently. The fourth configuration is designed to provide even more detailed information to the user. It enables two distinct analyses starting from an observation area by entering its radius in meters. The visive output is the same as the third configuration but considers the cumulative analysis of each point of the polygon as the second configuration. The tool performs a highly detailed visibility analysis in each analysis, providing the user with a comprehensive view of the observed area and enabling them to make more informed decisions. More information and detailed explanation of results can be found in section 3.3.

/	First Conf.	Second Conf.	Third Conf.	Fourth Conf.
Observation Point	YES	YES	YES	YES
Observation Radius	NO	YES	NO	YES
Analysis Radius 1	YES	YES	YES	YES
Analysis Radius 2	NO	NO	YES	YES
Elevation Model	YES	YES	YES	YES
Streets	YES	YES	YES	YES
Observer Height	YES	YES	YES	YES
Target Height	YES	YES	YES	YES

Table 3.1: Configurations Input Parameters

3.1.3 QGIS Plugin Interface

The following section explains the QGIS plugin interface, which allows users to obtain desired outputs by inputting the required parameters into designated fields. The process of opening a QGIS plugin interface involves several general steps. Initially, the user must install the plugin by navigating to the Plugins menu in the QGIS toolbar, selecting Manage and Install Plugins, and installing the desired plugin. Once installed, the next step is to activate the plugin by checking the box next to its name in the Manage and Install Plugins dialog. Following activation, a new menu or toolbar may appear in QGIS, providing access to the plugin's interface. Clicking on this menu or toolbar allows users to configure and use the plugin's features. Some plugins may necessitate additional configuration, requiring users to explore options or settings within the plugin interface. Upon opening the plugin, users can download it from the Processing tool screen and access the input screen, as seen in Figure 3.2a. The initial parameter required for successfully implementing the plugin is the DTM or DSM raster of the area of interest for analysis. This parameter serves as a crucial data source for the plugin's analysis. The Observation Point Coordinates field is the second parameter where users can enter the coordinates of the observation point. The Polygon Radius field is optional and defaults to zero, but it can be set to a specific value if the user desires to make observations from a particular area. This field controls the radius value in meters of the area where the observation will be made. The Analysis Radius 1 field is another crucial parameter, allowing users to determine the area to analyze based on the input value in meters. The Analysis Radius 2 field is optional, but it requires users to enter the value in meters of the radius of the area observed to compare two analyses. The Observer Height field and Target Height field are also essential, allowing users to set the observer's and target's height in meters above the ground, respectively. The Streets field is recommended to provide the plugin with the streets' vector layer to obtain accurate results. Figure 3.2b displays the possible outputs of the plugin. Once users have entered the necessary fields, they can save the outputs to a file or leave them as temporary files. The Streets output provides the vector layer with streets used in the algorithm, marked by a series of points. The second and third outputs are Visibility Analysis 1 and 2, allowing users to obtain two separate analyses of the streets around the observation area. The fourth output, named Area Analysis, generates a .tif file containing the analysis of visible and non-visible areas from the observation point enclosed within the radius set in the input parameters. Lastly, the Observation Points output generates the vector layer of the observation points used in the algorithm. The plugin comprehensively analyzes the area surrounding the observation point based on the input parameters.



(a) Part 1 **(b)** Part 2

Figure 3.2: Visibility Analysis Tool Input Parameters

3.2 Development

The GIS Desk Tool is a powerful plugin for QGIS that enables visibility analysis. It has been developed meticulously and with great attention to detail, using the Python programming language and the QGIS Desktop 3.32.0 software. QGIS allows data visualization using maps, charts, and diagrams while customizing the presentation with various symbology choices. For more complex geographical analysis, users can additionally use plugins and algorithms. QGIS also makes it simple to share and publish geospatial data as maps, online services, or print maps in various file formats, such as shapefiles, GeoTIFFs, and KML files. [8] Shapefiles are a standard geospatial vector data format used in GIS. They consist of multiple files storing both geometric and attribute data for map features. The main components include a .shp file for geometry, a .shx file for indexing, a .dbf file for attribute data, and a .prj file for projection information. GeoTIFFs are a raster image file format that includes geographic information embedded within the file. These files, based on the Tagged Image File Format (TIFF), are commonly used in GIS and remote sensing applications. The key characteristic of GeoTIFFs is the incorporation of georeferencing data, such as spatial coordinates and projection information, directly within the image file. KML, or Keyhole Markup Language, is an XML-based file format used for representing geographic data in three-dimensional Earth browsers, such as Google Earth and Google Maps. KML files can include a variety of data types, such as points, lines, polygons, images, and text annotations, allowing for the visualization of spatial information. Plugins are elements done to add functionality to the QGIS application. There are various plugins ready to be used and available to download. Plugins are fundamental to accomplishing complex

tasks and achieving results. The development process involved extensive research and experimentation, including studying existing plugins, using graphical modelers, and using the plugin builder in Python. The result is a robust tool that makes it easy to analyze visibility and perform a wide range of GIS tasks.

3.2.1 Plugin Studies

The first step in developing the tool was researching various existing QGIS plugins. The plugin considered for the study was the Visibility Analysis developed by Zoran Čučković. It includes four internal tools that can be used, namely: Viewshed Analysis, Intervisibility Network, Visibility Index, and Create Viewpoints.
Viewshield Analysis produces a visibility map showing visible and not visible zone. The output can be Binary viewshed, Depth below horizon, and Horizon. The basic output for viewshed analysis is a visibility map in raster format, which classifies the terrain surrounding an observation point into visible and not visible (true/false or 1/0). The logic behind this can be understood in Figure 3.3. The viewpoint traces rays that intersect with the terrain at specific points, allowing the distinction between the visible and non-visible areas. Depth below horizon will provide the depth at which lay invisible portions of a terrain. The value produced by this module can be understood as the theoretical height a construction should attain to appear on the horizon, as visible from the chosen observer point. Horizon option will trace a viewshed's outer edges, representing points that appear on the horizon from a chosen observer point. Intervisibility Network constructs a vector network of visual relationships between two sets of points (or within a single set). The depth below/above the visible horizon is also calculated for each link. The output of the intervisibility network routine is a network, in vector format, of visual relationships between two sets of points (or within a single set). For each link, the depth below/above the visible horizon is calculated, as in many cases only a portion of the specified target is visible. Visibility Index calculates the incoming/outgoing views for all-terrain locations. This module calculates the visual exposition of each data point for a given terrain model. The Index is calculated as the proportion of positive views, returning 1.0 when all views are positive. Create viewpoint is the first step in getting the observer points for the other three instruments already discussed. The observer points in the input will be processed and written as a geopackage file with standardized field names. Data will be reprojected to match the elevation model used if needed. Data inside the table can be changed manually - but the names and data types of fields should remain unchanged. [13]

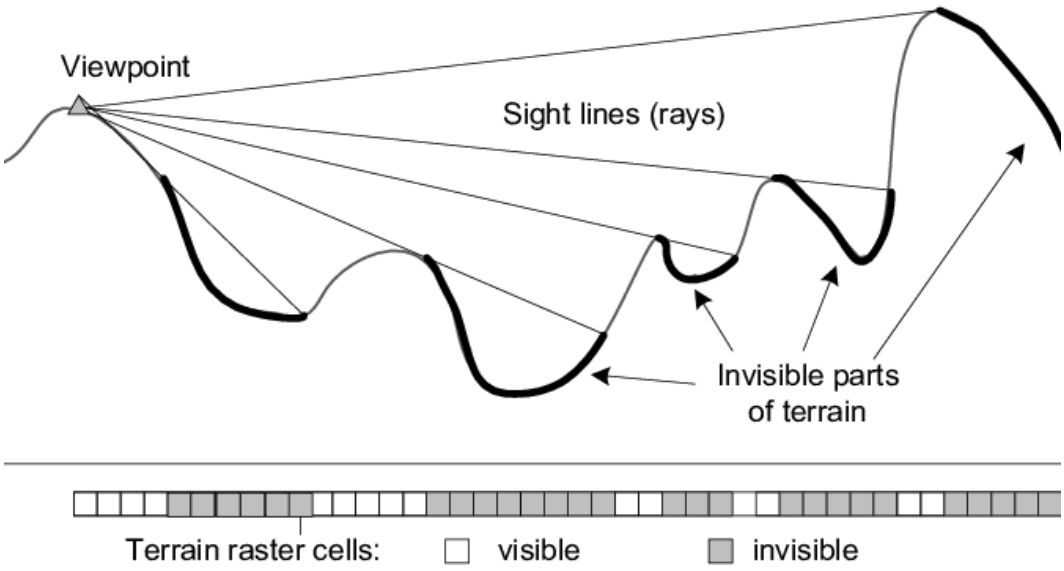


Figure 3.3: Viewpoint Logic [14]

3.2.2 Graphical Modeler

The second step in the process of developing the tool involved the use of the QGIS Graphical Modeler. It allows the user to create a visual structure for the plugin, enabling efficient organization. The graphical modeler allows the user to create complex models using a simple and easy-to-use interface. When working with a GIS, most analysis operations are not isolated, rather part of a chain of operations. Using the graphical modeler, that chain of operations can be wrapped into a single process, making it convenient to execute later with a different set of inputs. No matter how many steps and different algorithms it involves, a model is executed as a single algorithm, saving time and effort. [15] Figure 3.4 depicts the plugin structure developed using the graphical modeler. The yellow rectangular boxes represent the inputs to the model, which are the observation point, DTM/DSM, and streets. Initially, the observation point and streets were designed as vector layers, while the DTM/DSM input was in raster form. The white boxes represent various algorithms that deliver the desired result. Starting from the bottom of Figure 3.4, the "Create centroids along the line" algorithm generates a point vector from the input raster DTM/DSM and the linear vector Street. The algorithm creates points corresponding to the centroids of the pixels that intersect the linear vector, providing a punctual representation of the streets. The "Viewpoint" and "Streets Viewpoint" algorithms create a viewpoint for both the observation point and streets. Both algorithms take as input the raster DTM/DSM, Viewpoint takes the vector Observation Point, and Streets Viewpoint takes the "Create centroids

along the line" algorithm. The next step is "Viewshield," which creates a visibility map showing visible and non-visible areas using the raster DTM/DSM and the Viewpoint algorithm as input. The algorithm identifies which areas are visible and which are not. "Intervisibility Network" constructs a vector network of visual relationships between the input Viewpoint and Streets Viewpoint, taking them and the raster DTM/DSM as input. The algorithm identifies how objects are visible to each other. Finally, the green rectangular box shows the output of "Intervisibility Network" and "Viewshield." The output consists of an Intervisibility Network and a Viewshield, which help analyze and understand the visual relationships between objects in the model.

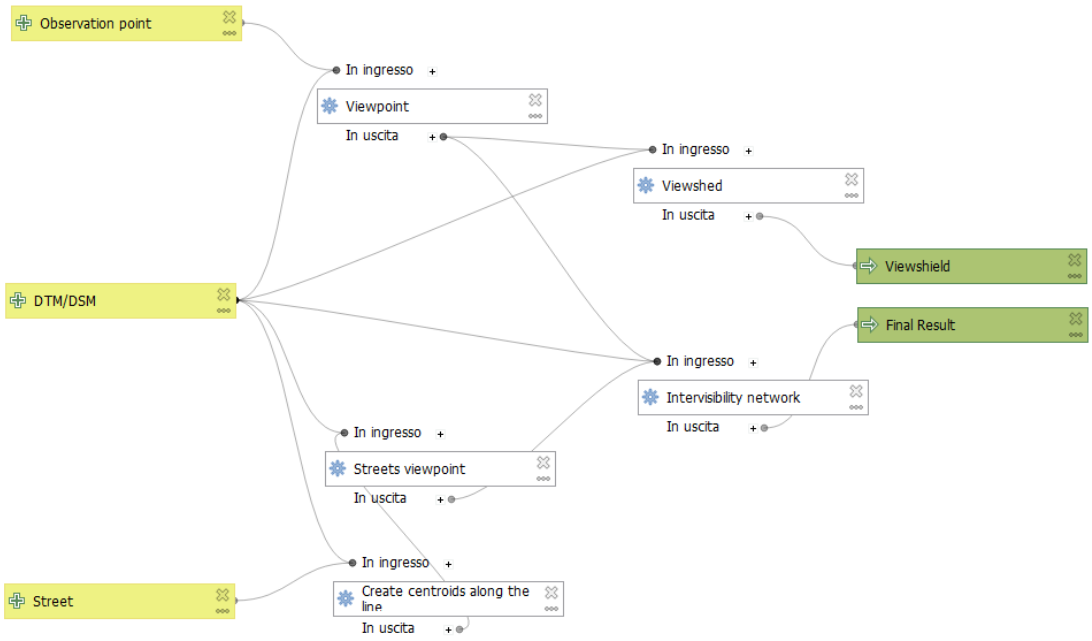


Figure 3.4: Initial tool structure

3.2.3 Code Development

Plugins are a great way to extend the functionality of QGIS. It is possible to write plugins using Python text editor ranging from adding a simple button to sophisticated toolkits. The final version of the visibility analysis tool has been implemented building a processing plugin for QGIS. This choice has been made because a process plugin is designed primarily for analysis, with user interaction limited to selecting inputs and outputs. The Processing Framework is essential to its development as it eliminates the need for a custom user interface, streamlining

the process. The built-in processing library generates a standard interface based on the inputs, resembling and behaving like any other processing algorithm in QGIS. Furthermore, it seamlessly integrates with the rest of the Processing framework, enabling the plugin algorithms to be utilized in batch processing, graphical modeler, and invoked from the Python console, among other functionalities. The development starts with a QGIS plugin named Plugin Builder, which creates all the necessary files and the boilerplate code for a plugin. Plugins Reloader is another helper plugin that allows the iterative development of plugins. This plugin helps change the plugin code and reflect it in QGIS without having to restart QGIS every time. To create a new plugin using Plugin Builder, it is necessary to fill out a form with all the relevant information, including the class name, plugin name, description, module name, version numbers, and author's credentials. Once this is done, it will be necessary to select the Processing Provider and output directory. Finally, it is possible to publish the plugin or classify it as experimental. It has been chosen as experimental because of the academic purposes of this thesis. The code development step can be found into the plugin directory and in the selected code file in Python text editor. In Appendix A, it is possible to find the plugin code. The code starts with the libraries import in section A.1.1. The line "from qgis.PyQt.QtCore import QCoreApplication" is importing the "QCoreApplication" class from the "QtCore" module of the "PyQt" package in the QGIS Python Application Programming Interface (API). In particular, "qgis" is the QGIS package providing access to the QGIS API for Python scripting. "PyQt" is a set of Python bindings for the Qt application framework, and QGIS uses it to create its Graphical User Interface (GUI) and other functionality. "QtCore" is a module within PyQt that provides core non-GUI functionality and includes essential classes and functions for event handling, file I/O, data types, and more. "QCoreApplication" is a class in the "QtCore" module. It represents the core application object in Qt and provides functionality related to application-wide resources, event handling, and more. In QGIS scripting, importing "QCoreApplication" is often necessary to ensure the script runs within the Qt application framework. This is important for handling events and ensuring proper integration with the QGIS environment. The following line indicates that the code imports specific classes or modules from the QGIS core module. The various libraries imported are:

- QgsProcessingAlgorithm: This class represents a processing algorithm in QGIS. It is the base class for all processing algorithms.
- QgsProcessingParameterNumber: This is a parameter type for numerical input. It is often used when the algorithm requires numeric values as input.
- QgsProcessingParameterRasterLayer: This parameter represents a raster layer in QGIS. It is used when the algorithm needs a raster layer as input.

- QgsProcessingParameterPoint: This parameter type represents a point in QGIS. It can be used when the algorithm requires a point as input.
- QgsProcessingParameterMapLayer: This parameter represents a map layer in QGIS. It is used when the algorithm needs a generic map layer as input.
- QgsProcessingParameterFeatureSource: This parameter represents a feature source in QGIS. It is used when the algorithm requires a vector layer (feature source) as input.
- QgsProcessingParameterFeatureSink: This parameter represents a feature sink in QGIS. It is used when the algorithm produces a vector layer as output.
- QgsProcessingParameterRasterDestination: This parameter represents a destination for raster data. It is used when the algorithm produces a raster layer as output.
- processing: This is the QGIS processing module, providing access to various geospatial processing algorithms.

The section A.1.2 starts with the definition of the class "VisibilityAnalysisToolAlgorithm" as a custom class that extends the functionality provided by the "QgsProcessingAlgorithm". The section follows with the parameters initialization method. The parameters defined are the following:

- DTM or DSM: A raster layer representing a Digital Terrain Model (DTM) or Digital Surface Model (DSM). It is an input parameter that allows the user to select a raster layer.
- Observation Point Coordinates: It is an input parameter point that allows the user to specify the location of the observation point.
- Polygon Radius: A numeric optional parameter representing the polygon radius in meters with a default value of 0.
- Analysis Radius 1: A numeric parameter representing analysis radius 1 in meters with a default value of 10,000. m.
- Analysis Radius 2: Similar to Analysis Radius 1 but for Analysis Radius 2. It is optional and has a default value of 0.
- Observer Height: A numeric parameter representing the observer's height in meters. It has a default value of 1.6 meters.
- Target Height: A numeric parameter representing the target height in meters with a default value of 0.

- Streets: A vector map layer representing streets. It is an input parameter that allows the user to select a vector layer containing streets.
- Streets: A vector feature sink for streets. It is an output parameter that stores the processed street data.
- Final Result Analysis 2: A vector feature sink for the final result of Analysis 2. It is an output parameter that stores the processed data for Analysis 2.
- Final Result Analysis 1: A vector feature sink for the final result of Analysis 1. It is an output parameter that stores the processed data for Analysis 1.
- Area Analysis: A raster destination representing the area analysis. An output parameter stores the analysis result as a raster layer.
- Observation Point: A vector feature sink for observation points. It is an output parameter that stores the processed observation points data.

Next to the parameters definition is the method "processAlgorithm," which is the main entry point for all plugins' processing logic. In the following line, there is the object "QgsProcessingMultiStepFeedback" creation. This object is used to manage the feedback during the algorithm's execution. It is handy when the algorithm consists of multiple steps, and it is defined to report progress at each step. "results" and "outputs" are variables respectively defined to store intermediate results or data during the algorithm's execution and to store the final outputs of the algorithm. Subsequently, the algorithms part comprehends the plugin's core with all the logic. A brief summary of the input parameters and algorithms is depicted in Figure 3.5. The first algorithm is "qgis:generatepointspixelcentroidsalongline" and can be found in A.1. It generates points along a line by calculating the pixel centroids taken from the DTM or DSM raster over the Streets vector in the entrance. The output is set to "QgsProcessing.TEMPORARY OUTPUT", meaning the result will be temporarily stored in the project's temporary directory.
The second algorithm is "native:pointtolayer" and can be found in A.2. His role is to convert the input point to a point layer taken as input the written coordinates of the observation point. His output point layer is stored in the results dictionary under the key "Si", allowing it to be shown on the map.
The third step of the algorithm involves the "native:buffer" operation, which is detailed in A.3. This operation is designed to create a buffer around the observation point when the specified buffer distance is not equal to zero. The buffer is computed based on the provided distance, forming a protective zone around the observation point. It takes as input the result of the previous algorithm, the layer observation point. The resulting buffer layer is stored in the outputs dictionary under the identifier "Buffer".

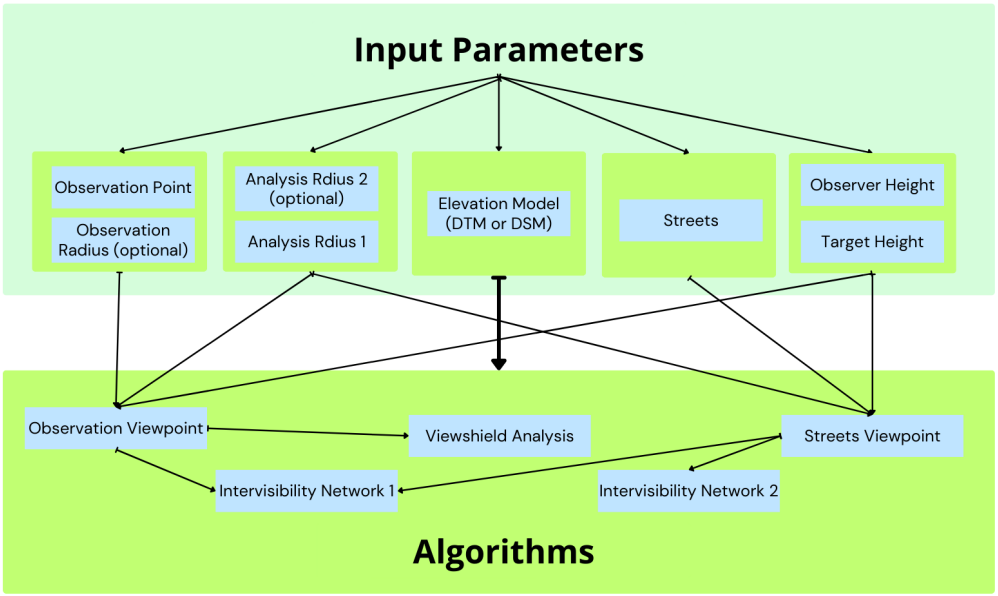


Figure 3.5: Development Scheme

The next step involves the "native:generatepointspixelcentroidsinsidepolygons" algorithm, described in A.4. This operation generates points inside the polygon created in the previous step. It utilizes the DTM or DSM specified by the user and the polygon layer generated in the buffer operation. The resulting point layer is stored in the outputs dictionary under the key "GeneraPuntiCentroidiDeiPixelDentroPoligoni", allowing it to be shown on the map.
The fifth step introduces the "visibility:createviewpoints" algorithm, outlined in A.5. This algorithm is responsible for generating observation points based on specified parameters. It utilizes the DTM or DSM provided by the user and the point layer generated in the previous step, which could be a single point or a composition. It also utilizes the target height and observer height as input. The resulting observation points are stored temporarily and can be referred to in subsequent stages of the algorithm.
Similar to the fifth algorithm, the next one utilizes the "visibility:createviewpoints" algorithm to generate visibility analysis points along a line, incorporating input parameters such as observer height, target height, analysis radius, DTM or DSM, and the output of algorithm seen in A.1. The resulting points are stored in the specified vector layer "Strade", allowing it to be shown on the map.
The sixth algorithm A.7 uses the "visibility:viewshed" algorithm to perform a viewshed analysis. It considers parameters such as the type of analysis (binary viewshed), the DTM or DSM, observer points as the output of A.5 algorithm,

and additional settings like the refraction coefficient and curvature. The resulting viewshed analysis output is stored in the specified raster layer "Analisi". The generated layer is accessible in the outputs dictionary under the key "Viewshed", allowing it to be shown on the map.
The seventh algorithm uses the "visibility:intervisibility" algorithm to create an intervisibility network, as shown in A.8. The algorithm considers parameters such as the DTM or DSM, observer points as the output of A.5, target points derived from the visibility analysis along streets ("Str"), and settings like the refraction coefficient and curvature. The resulting intervisibility network is stored in the specified vector layer "FinalResult". The generated layer is accessible in the outputs dictionary under the key "IntervisibilityNetwork", allowing it to be shown on the map.
Algorithms eight, nine, and ten, respectively A.9, A.10 and A.11 are the same as A.5, A.6 and A.8 with the condition that they are executed if Analysis Radius 2 is set, and the user want output with two analysis.
Subsequently, some methods are mandatory for the plugin implementation. The name definition method in section A.1.3 returns the algorithm name used for identifying the algorithm. DisplayName definition method in section A.1.4 returns the translated algorithm name, which should be used for any user-visible display of the algorithm name. The group definition method in section A.1.5 returns the name of the group to which this algorithm belongs. GroupID definition method in section A.1.6 returns the unique ID of the group this algorithm belongs to. The try definition method in section A.1.7 is used for translating text to the user's language. Create instance definition method in section A.1.8 is responsible for creating an instance of the algorithm class.

3.3 Results

3.3.1 Outputs

The output generated by the GIS Desk tool can be visualized through QGIS. As elucidated in section 3.1, the plugin affords four configurations that produce distinct outputs. Table 3.2 displays the input parameter values for the four configurations. The Observation point is the same for all four configurations and is expressed in Cartesian coordinates (27229.9E, 5070829.2N). When specified, the Observation Radius is set to 15 meters. Analysis Radius 1 is 10000 meters, while Analysis Radius 2 is 4000 meters when specified. Observer Height and Target Height are consistently set to default values of 1.6 and 0 meters, respectively. Elevation Model and Streets remain the same across all configurations. Figure 3.6 shows the first configuration called also single observation point and single radius configuration. This setup necessitates the observation point coordinates, the radius in meters,

the DTM, the target height, the observer height, and the roads as inputs. The legend of figure 3.6 clarifies the output's constituents. The red dot represents the observation point, while the purple lines denote the roads used in the analysis. The blue segments originate from the observation point and extend to the visible roads. The DTM employed in the analysis is presented, and two colors represent the binary visibility analysis. The visible portion is presented transparently, while the non-visible portion is depicted in black. Figure 3.7a and figure 3.7b provide a more detailed look at the information presented in Figure 3.6. In particular, Figure 3.7a highlights the observation point in red and shows all the blue segments that originate from this point. These blue segments represent the visible areas from the observation point, a crucial aspect of the intervisibility network. On the other hand, figure 3.7b provides more information about the endpoints of the value segments in the intervisibility network. Additionally, it shows the purple points that depict the streets. The borders in this figure define the visibility area, with the blue segments extending only as far as the points along the purple street outside the black area. These figures help clarify the plugin's functionality and provide a more detailed understanding of the intervisibility network. By examining these figures closely, it is possible to appreciate better the visible areas and those that are not. Figure 3.8 shows the second configuration possible called also single observation point and two rays configuration. Observation point coordinates, two rays in meters, the DTM, the target height, the observer height, and the roads are taken as inputs. The difference with 3.6 is the second output of the intervisibility network in green. The utility of having two kinds of analysis on the streets is evident here. Figures 3.9, 3.10a, and 3.10b provide an overview of the third configuration, called also observation area and single radius configuration. Figure 3.9 displays the outcome of the visibility analysis carried out from an observation area with a radius of 15 meters. Figure 3.10a shows the four points that define the observation area from which the blue segments of the intervisibility network originate. The cumulative analysis of the entire area has been employed better to understand the appropriate size for an observation area. This approach allowed the user to comprehend the observation area's size better and optimize it for specific applications. Figure 3.11 displays the same output as figure 3.9, but with two intervisibility networks, one in green and one in blue. This represents the fourth configuration, called also observation area and two rays configuration. Its usage is intended for specific situations where a detailed response is required. Similar to the second configuration, these two analyses provide the user with a more comprehensive understanding.

/	First Conf.	Second Conf.	Third Conf.	Fourth Conf.
Observation Point	(x,y)	(x,y)	(x,y)	(x,y)
Observation Radius		15 m		15 m
Analysis Radius 1	10000 m	10000 m	10000 m	10000 m
Analysis Radius 2			4000 m	4000 m
Elevation Model	YES	YES	YES	YES
Streets	YES	YES	YES	YES
Observer Height	1.6 m	1.6 m	1.6 m	1.6 m
Target Height	0 m	0 m	0 m	0 m

Table 3.2: Configurations Input Parameters (x,y) = (427229.9E,5070829.2N)

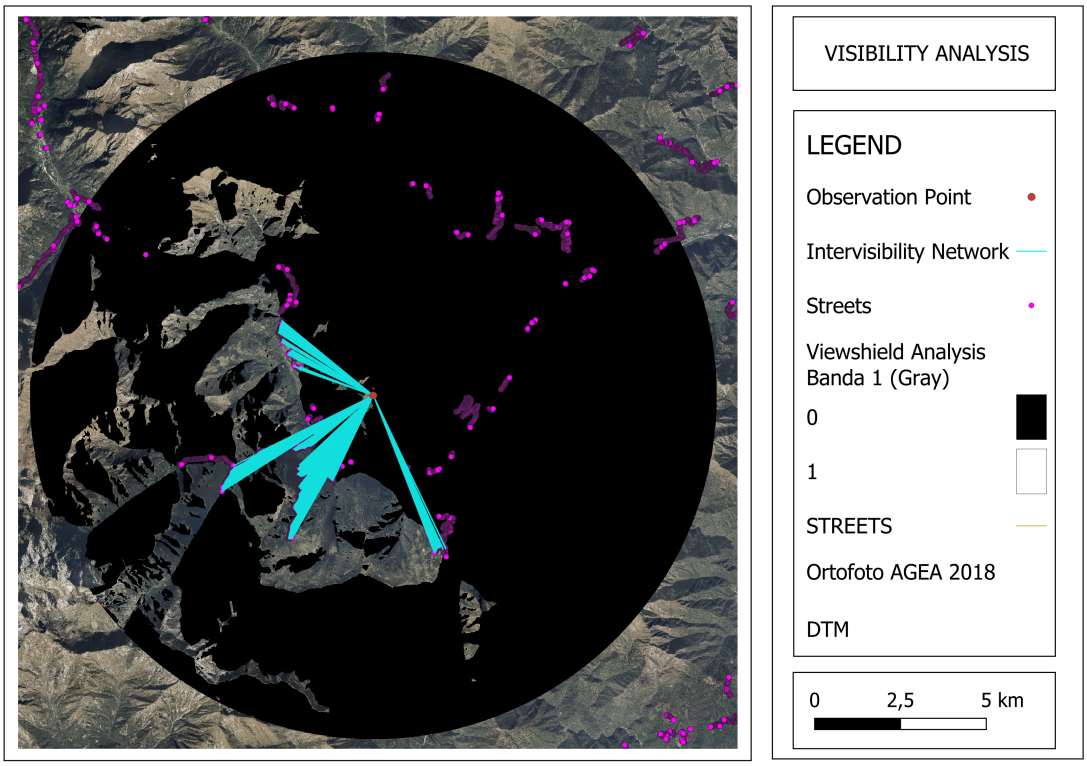
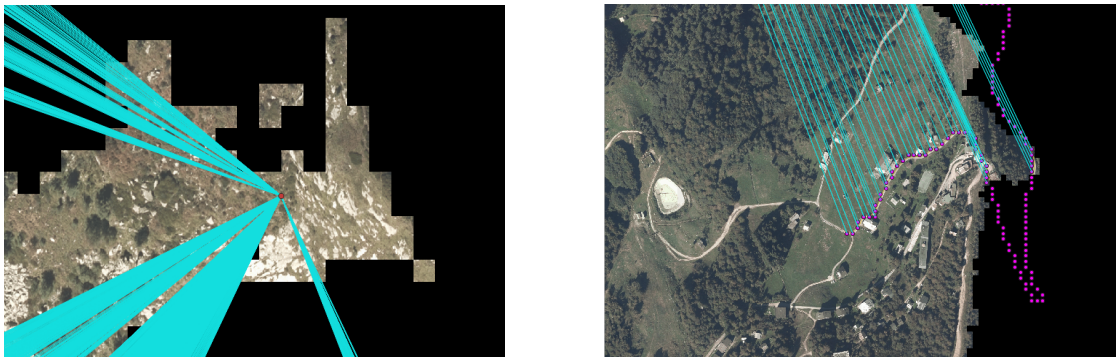


Figure 3.6: Algoritm result on single observation point and single radius configuration



(a) Single Observation Point **(b)** Segment Terminal on Street

Figure 3.7: Detail on result 3.6

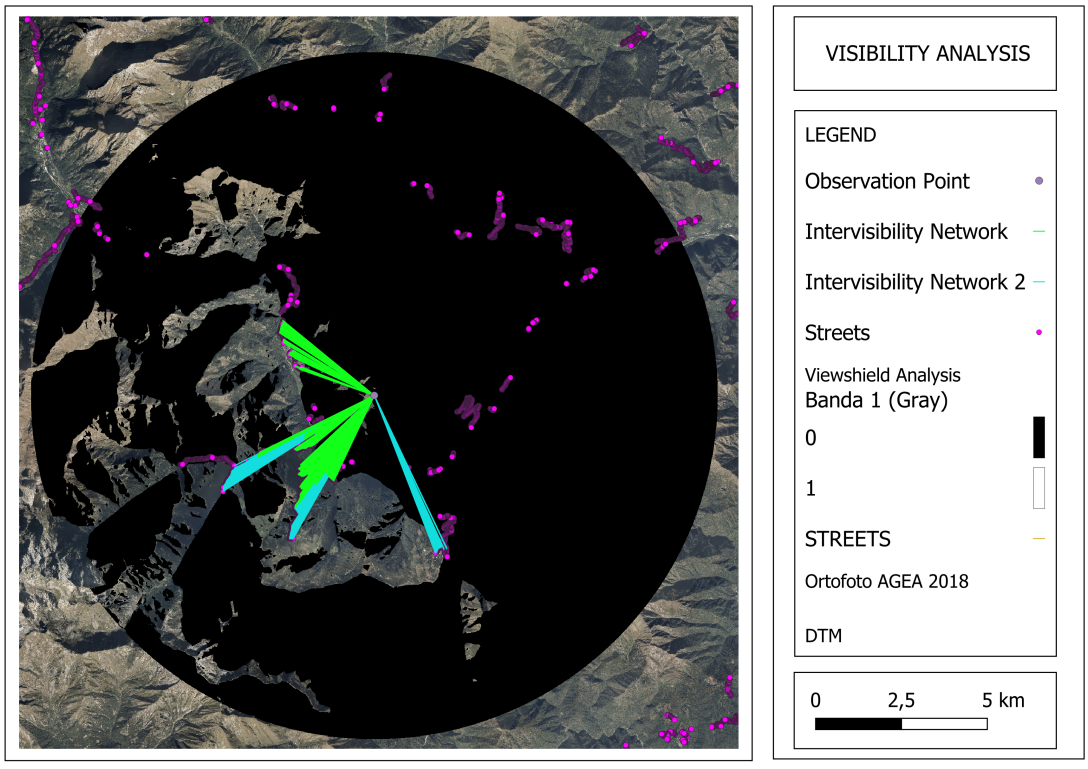


Figure 3.8: Algoritm result on single observation point and two rays configuration

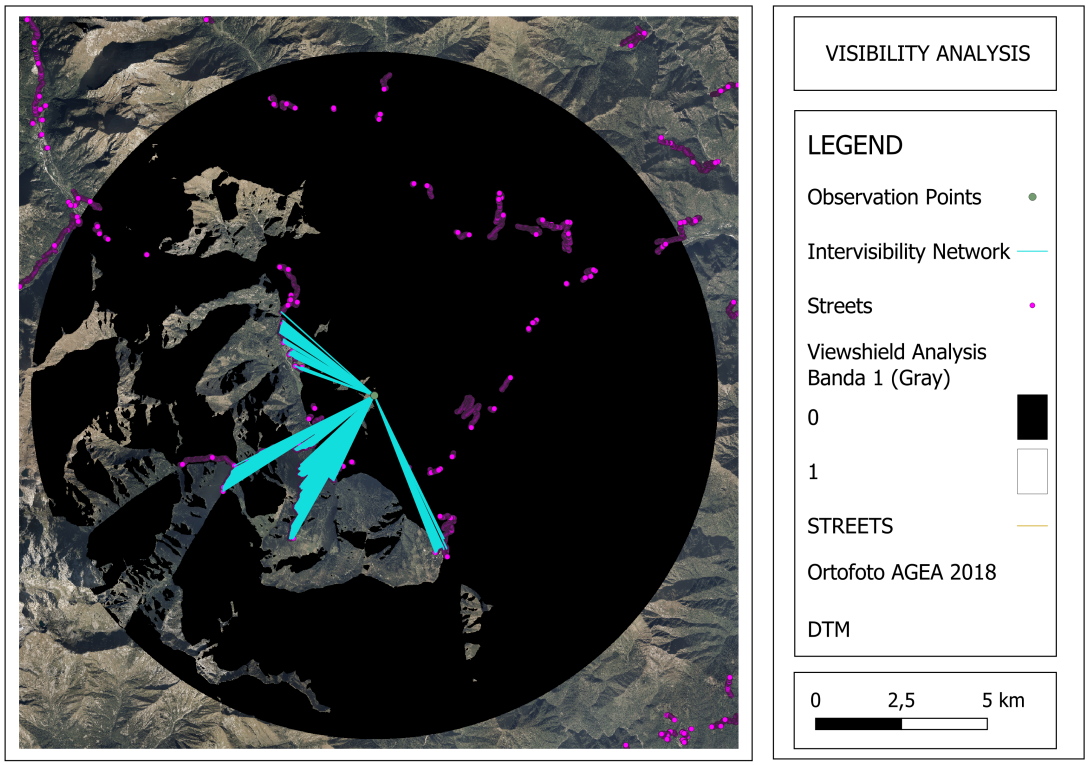
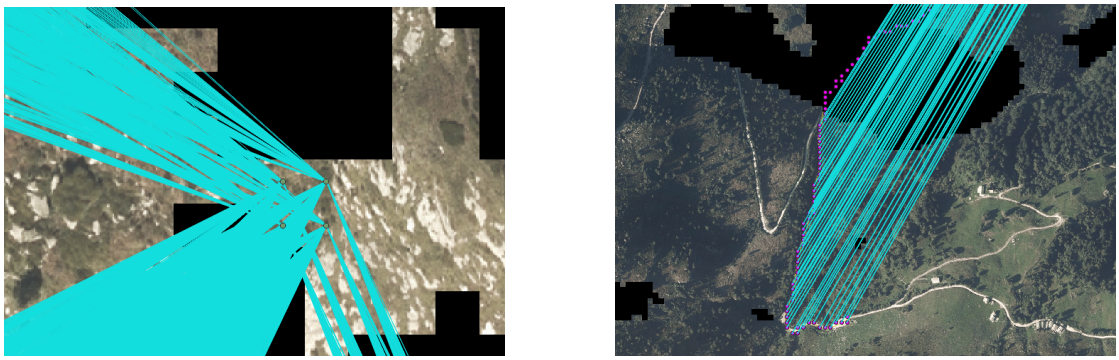


Figure 3.9: Algoritm result on observation area and single radius configuration



(a) Observation Area Points

(b) Segment Terminal on Street

Figure 3.10: Detail on result 3.9

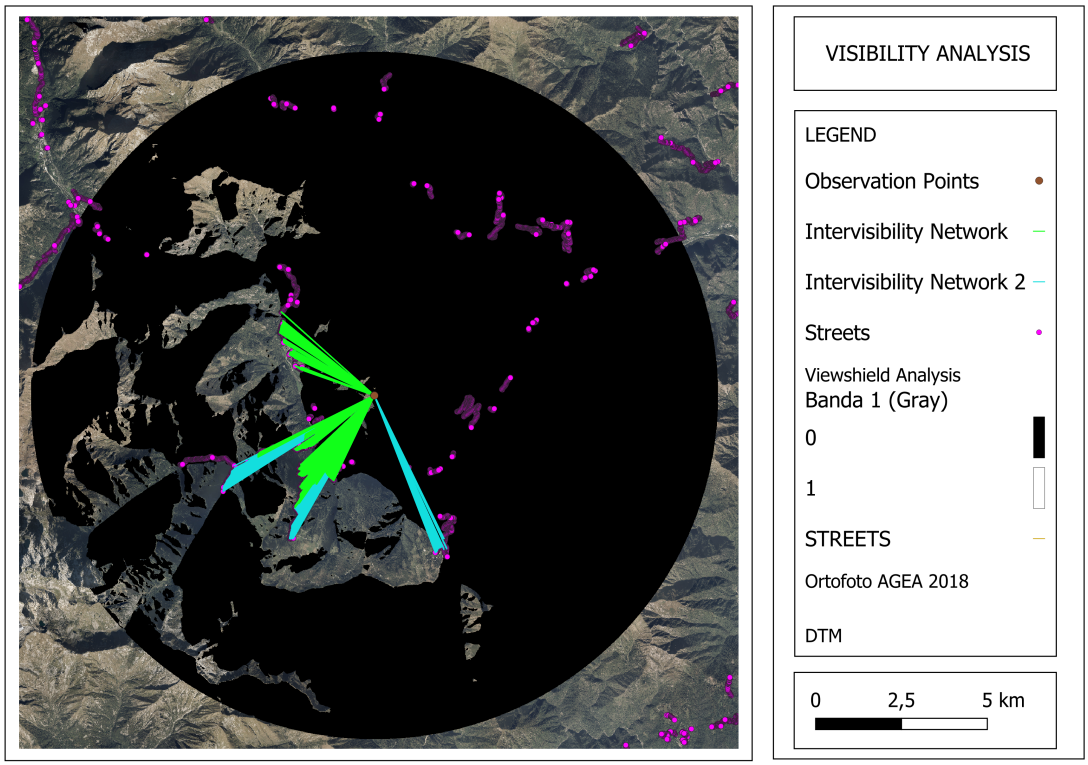


Figure 3.11: Algoritm result on observation area and two rays configuration

3.3.2 Output Analysis

This section delves into the results obtained from the algorithm and elucidates the correlation between the input parameters and the algorithm's execution time. The algorithm's execution time is a crucial metric in achieving the set objectives and directly influences the planning phase. Inadequate time allocation during this phase can result in suboptimal outcomes. To better understand how the input parameters affect the algorithm's performance, the execution time will be analyzed for all four possible configurations with different input parameters. Execution time values are considered as the mean values of several tests done. This analysis will provide insight into the input parameters that significantly impact the execution time and those that do not. Identifying the critical input parameters makes optimizing the algorithm's performance and setting limits possible. The first configuration under examination involves a single observation point as input, with area analysis radius variations. The graph 3.12 clearly illustrates the trend of execution time with six different input rays. The first analysis radius is 5000 meters, a vast range allowing a complete view. The plugin's execution time is just under 20 seconds, an excellent result in terms of planning. The most extensive analysis is conducted

with a radius of 30000 meters, a comprehensive analysis that requires specific considerations for its use. This analysis requires around 7 minutes, which is a tolerable time compared to the significant extension. Graph 3.12 shows that the times remain around a minute up to 15000 meters but spike at 20000 meters radius. This favorable outcome covers significant distances in a minute or less. The second configuration under examination involves a single observation point as input, with two different area analysis radius variations. The graph 3.13 clearly illustrates the trend of execution time with six different input rays. The first two analysis rays are 5000 and 10000 meters, a vast range allowing a complete view. The plugin's execution time is just around 50 seconds, an excellent result in terms of planning. The most extensive analysis is conducted with two rays of 30000 and 25000 meters, and it required around 10 minutes, which is a tolerable time compared to the significant extension. Graph 3.13 shows that the times remain under three minutes up to 15000 meters but spike at 20000 meters radius. This outcome is coherent with the previous analysis considering only one radius. The third configuration under examination involves multiple observation points as input in a radius of 25 m, with area analysis radius variations. The graph 3.14 clearly illustrates the trend of execution time with five different input rays. The first analysis radius is 5000 meters, a vast range allowing a complete view. The plugin's execution time is just around 40 seconds, an excellent result in terms of planning. The most extensive analysis is conducted with a radius of 25000 meters, and it requires around 4 hours, which is a tolerable time compared to the significant extension. However, compared to other configurations, his time is higher. Graph 3.13 shows that the times remain around seven minutes up to 15000 meters but spike at 20000 meters radius. The outcome is coherent with the previous analysis and evidence that staying within 15000 meters for rapid request analysis is better. The fourth configuration under examination involves multiple observation points as input in a radius of 25 m, with two different area analysis radius variations. The graph 3.15 clearly illustrates the trend of execution time with four different input rays. The first couple of analysis radius is 5000 and 10000 meters, a vast range allowing a complete view. In that case, the plugin's execution time is just around 3 minutes, an excellent result in terms of planning. The most extensive analysis is conducted with two rays of 20000 and 25000 meters, and it required around 7 hours, which is a tolerable time only in the planning phase. Graph 3.15 shows that the times remain under 10 minutes up to 15000 meters but spike at 20000 meters radius. This outcome is coherent with the previous analysis considering only one radius.

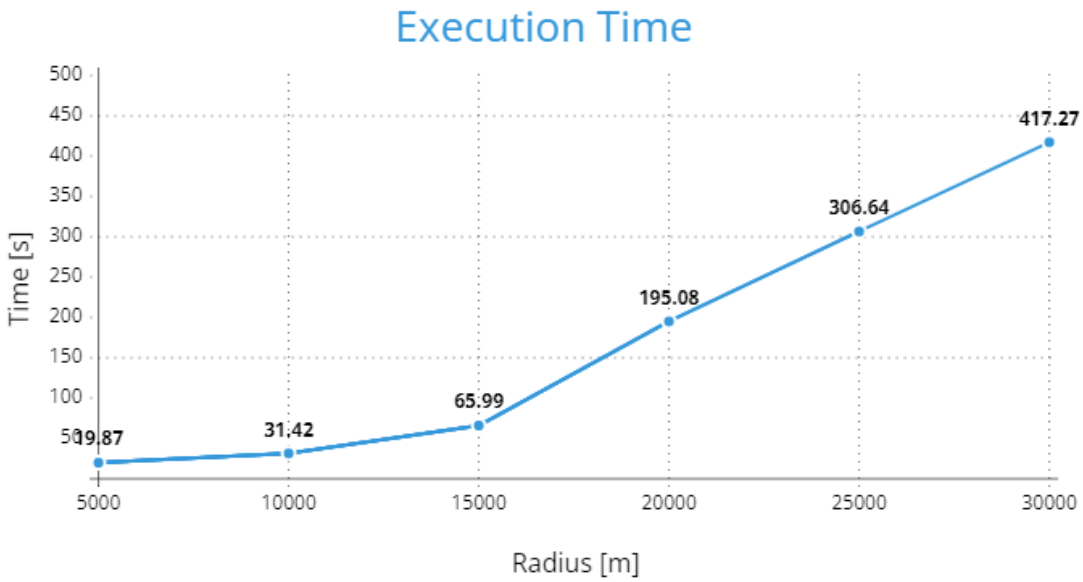


Figure 3.12: Execution time for first configuration

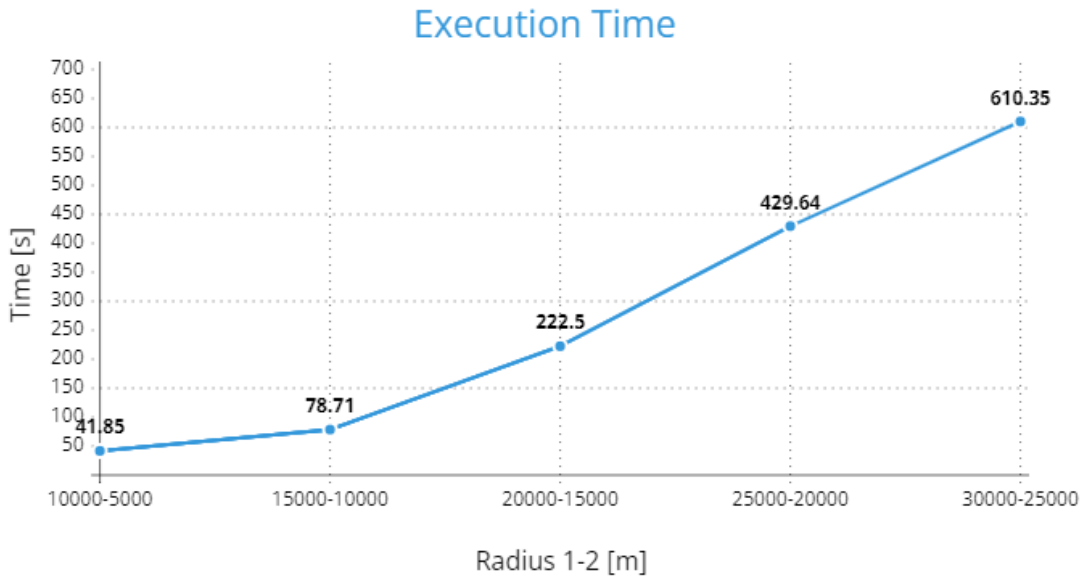


Figure 3.13: Execution time for second configuration

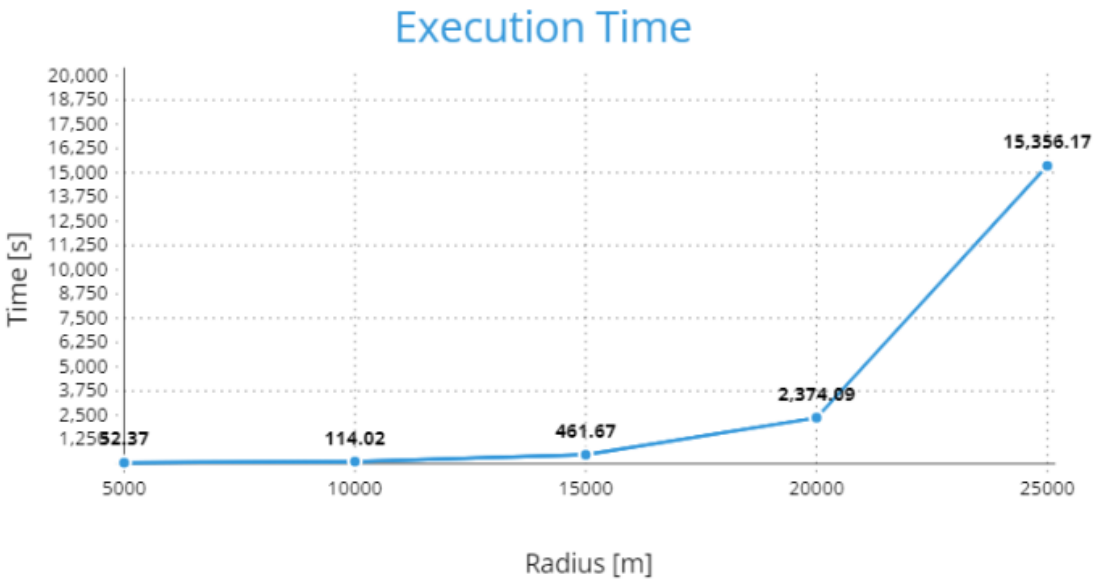


Figure 3.14: Execution time for third configuration

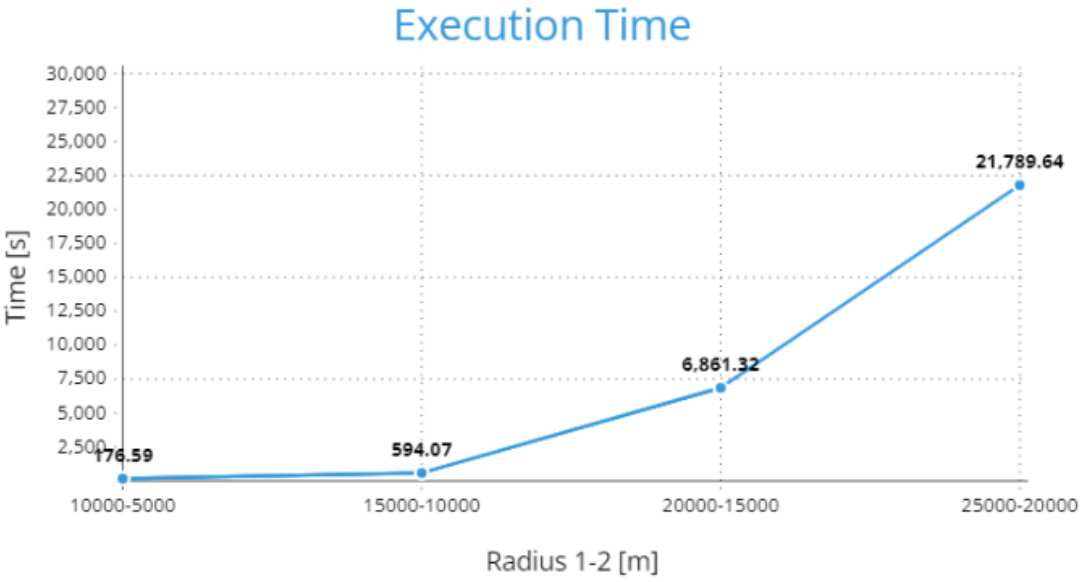


Figure 3.15: Execution time for fourth configuration

3.3.3 Limitations

Upon analyzing the results generated by the GIS Desk tool, it is fundamental to delineate the limitations encountered. These limitations may manifest in various forms, ranging from input values that significantly impede algorithmic execution to constraints that necessitate the impracticability or redundancy of the algorithm. Exploring these limitations provides critical insights into the tool's applicability in specific contexts. This approach is vital in creating solutions better suited to end user needs. The first limitation is the analysis radius, which cannot exceed 30000 m. This is usually sufficient for most applications, but it can be restrictive for assessments that require analysis beyond this range. Additionally, the tool may take longer to process data when analyzing long distances and using two-ray visibility. In some cases, processing times can extend to several hours, affecting the tool's responsiveness. Another tool limitation involves the observation radius when multiple observation points are used. As the observation area radius increases, processing time increases significantly, especially when set above 25-30 m. This limitation aligns with the strategic nature of stakeouts, where the extension has a practical limit. The last limitation concerns data availability. Before using the GIS Desk tool, it is necessary to prepare streets, Digital Terrain Models (DTM), or Digital Surface Models (DSM) in advance. In some cases, it may be necessary to extract or create this data for areas that lack it. This requirement adds a preparatory step to the tool's workflow and limits its adaptability to areas with pre-existing geospatial data. These limitations highlight the importance of considering practical constraints, processing times, and data availability when using the GIS Desk tool in different contexts.

Chapter 4

GIS Mobile Tool

The GIS Mobile Tool is a highly versatile tool that is designed to cater to a wide range of needs and to meet specific expressed requirements. It has been designed to be combined with the GIS Desk Tool or act independently to meet some requirements that still need to be satisfied. The mobile tool allows the transmission of input data from an application and receives visibility analysis as output. It aims to address various challenges, such as the lack of on-site information and time constraints. Its significant advantage is the ability to send data to a distant and fixed station, eliminating the need to preload data as they are already present in the fixed station's database. Another benefit is the variable response time, allowing for a quick response to an unplanned request. The tool requires an internet connection to function, but when combined with the desk tool, it can effectively address all the previously mentioned challenges.

4.1 Functionalities

The GIS Mobile Tool represents an application designed for smartphones or desktop devices, structured to transmit data to a fixed station and receive a detailed image of the visibility analysis for a specific area as output. Visibility analysis can be conducted intuitively, starting simply from a point identified by its coordinates. The user interested in performing such an analysis selects a reference point and specifies the radius of the area of interest, obtaining a detailed visual result. Thanks to specialized algorithms, the visual output consists of an image containing a map of the considered area, enriched with numerous segments extending from the user-entered point to the surrounding roads. The presence of segments reaching the roads indicates the visibility of the road from that point, while the absence of segments indicates the area is not visible from that perspective. Additionally, the user has the option to enter a third piece of information, represented by the

radius in meters of the area from which they want to initiate the analysis. The entered coordinates constitute the center of this area that is created. In this case, a combined analysis of the surrounding area is obtained, considering various points within the area defined by that radius. This tool streamlines the submission of essential data, such as the coordinates of the point of interest, the analysis radius, and, optionally, the radius of the area for conducting an analysis from an area rather than a specific point. Using this data, along with various elevation models and files containing information about the surrounding roads, stored in a database, the application can perform a visibility analysis of the area without requiring complex initial preparation. As mentioned earlier, the convenience of being able to use this tool directly on one's mobile device allows for obtaining a detailed analysis in just a few minutes.

4.1.1 Flow Chart

The tool is primarily divided between the front-end and back-end. The front-end consists of three distinctive pages: one for login, one for data submission, and one for viewing the resulting image. The login page interacts with the back-end, transmitting user credentials. The back-end connects to the database to verify the congruence of the entered data. In case of a match, the back-end communicates with the front-end, authorizing access. The data submission page offers dedicated fields for plugin parameters, including the coordinates of the observation point, the analysis radius, and, optionally, the area radius for conducting an analysis from an area rather than a point. Once the data is entered, the front-end transmits it to the back-end, which stores it in the database. The third page focuses on the visualization of the image and the result of the data submission. When the result is requested, the front-end forwards the request to the back-end, which leverages a batch file and a QGIS algorithm. The back-end initiates the batch file to configure the QGIS environment and launch an algorithm based on the visibility analysis algorithm described in the previous chapter's GIS Desk Tool. The algorithm retrieves the necessary data from the database and proceeds with the visibility analysis. Once completed, the back-end processes the results and sends a response to the front-end, enabling the image's display through the application. It is worth noting that the database is divided into two parts: SQLite, containing manually entered data from the application, and a second database with all DTM or DSM and road files useful for analysis. Both serve as inputs for the QGIS algorithm. Figure 4.1 provides a detailed view of the tool's flow chart. Here it is possible to see how credentials are entered for the login page in the dark green box. After checking the credentials in the database, a response is returned. In the case of incorrect credentials, access to the next page is denied. In the case of correct credentials, access is gained to the data entry page. Here, it is possible

to log out and return to the login page or enter data. It is also possible to insert some data inside some white box. This data includes, as previously mentioned, the coordinates of the observation point, the analysis radius and, optionally, the radius of the area to conduct an analysis from an area rather than a point. This data is entered into the database. Next, it is possible to pass to the visibility analysis page. Here the analysis is requested and the actual algorithm is executed. A batch file that sets up the QGIS operation environment is executed and the algorithm is started. The algorithm takes data from the two databases: the one in which the data was previously entered and a database containing roads and the elevation model of the area. Having all the inputs, the algorithm produces a KML file of the Intervisibility Network. This is converted into an image and overlaid on a map of the area. Once the process is complete, the output is displayed as an image on the visibility analysis page of the application.

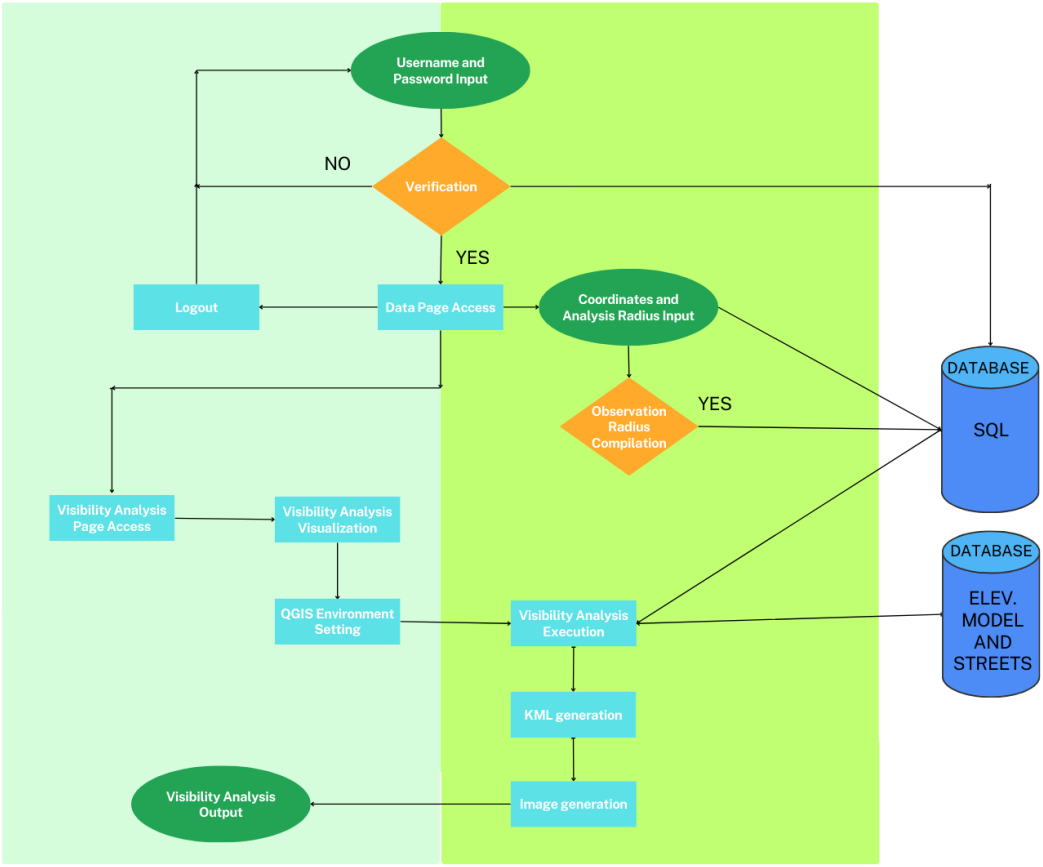


Figure 4.1: GIS Mobile Tool Flow Chart

4.1.2 Front-End

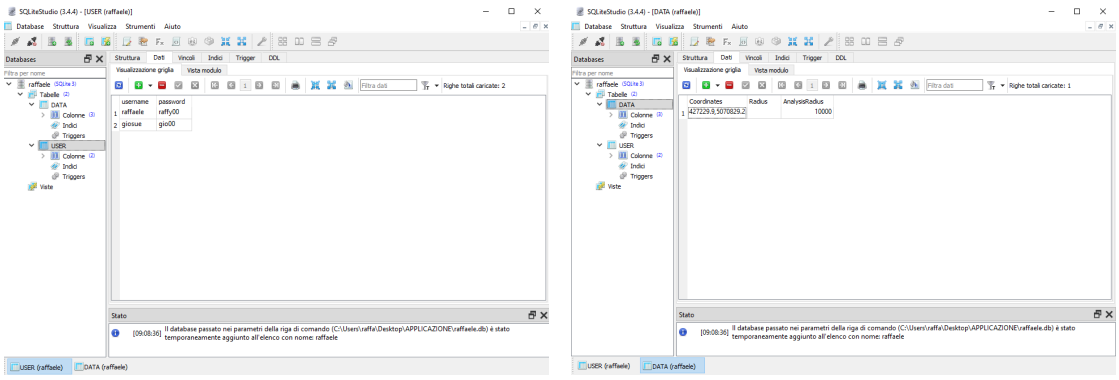
The front-end of the tool has been entirely developed using the Flutter framework. This framework provides the ability to organize code into classes, ensuring a flexible and user-friendly structure. In our implementation, we have adopted a specific architecture for creating the three previously mentioned pages, each with its associated classes. The choice of Flutter has allowed us to achieve a versatile front-end that is highly adaptable to the tool's requirements. The class-based organization facilitates code management and maintenance, improving readability and enabling rapid development. In the developed code, each of the three pages has been carefully designed to perform specific functions. Communication between the front-end and back-end occurs efficiently through the Representational State Transfer (REST) protocol. In this communication model, the front-end acts as a client, interacting with the back-end through RESTful requests and responses. This client-server architecture ensures reliable and fast communication between the components of the system. The front-end sends requests to the back-end, which processes these requests and returns the necessary responses to ensure a smooth and consistent user experience. Overall, the use of Flutter for front-end development provides a robust and efficient environment, ensuring a solid technological foundation for the tool's functionality.

4.1.3 Back-End

The back-end has been entirely developed using the Python programming language. Its primary function is to orchestrate the operation of the application by acting as a server, managing and responding to all requests coming from the client. It interacts with both the SQLite database and a dedicated bash file to configure the QGIS environment. The bash file is a shell script, a command program used in Unix and Unix-like operating systems. In this context, the bash file is essential for setting up the QGIS environment. This procedure is necessary because QGIS requires a dedicated environment to operate in stand-alone mode, and the bash file is responsible for preparing and launching this environment. The back-end can execute the bash file on request, which sets up the necessary environment for QGIS to operate in stand-alone mode. This critical phase is activated only when requested by the client's output, ensuring a safe and controlled approach. The interaction between the algorithm and the database plays a crucial role, enabling efficient and rapid data transmission. This dynamic connection between the back-end, the bash file, and the database ensures the consistent functioning of the application, providing the client with precise and timely results. In short, the back-end is the powerhouse behind the application's operations, effectively managing the complex interactions among various components, including the preparation of the QGIS environment through the bash file.

4.1.4 Database

The database used for the entire plugin has been created with SQLite, chosen for its ease of use and versatility as a database management system. The composition of the database is illustrated in figure 4.2. In particular, in figure 4.2a, it is possible to examine the structure of the user table, which consists of two columns containing credentials for users authorized to access the plugin. To ensure the uniqueness of users, a primary key has been set on the username column. In figure 4.2b, the data table is highlighted, characterized by three columns for point coordinates and two columns for the rays related to visibility analysis. This table is essential for storing crucial information needed for algorithm execution and analysis generation. The creation of an SQLite database involves defining tables and columns. A table represents an organized set of data, while columns define the different categories of information in the table. In the context of our plugin, the user table and the data table are examples of tables, while the columns represent various pieces of information, such as username and password for the user table, and point coordinates and radii for the data table. Using a primary key for the username in the user table is a common practice to ensure the unique identification of users, avoiding unwanted duplications. This primary key plays a key role in the database structure, ensuring the integrity and consistency of the stored information.



(a) User **(b)** Data

Figure 4.2: Database

The tool makes use of an additional database that needs to contain numerous elevation models relating to the area in which an analysis might be required. In addition, it must include vector layers with the roads in the zones mentioned above. The size and completeness of this database play a fundamental role, directly influencing the accuracy and completeness of the results obtained during the analysis. The accurate management of this extensive database is essential to guarantee the quality and reliability of the analysis conducted.

4.1.5 Tool Interface

The three pages of the tool interface can be observed in figures 4.3, 4.4, and 4.5. This desktop representation is provided to facilitate the visualization of the various components of the tool but is also adapted for use on mobile devices. The use on mobile device is shown in figure 4.6. The login page, depicted in figure 4.3, is clear and functional with two fields for entering credentials and a login button. Clicking on this button leads to the second page, dedicated to parameter input, as illustrated in figure 4.4. This page features three empty boxes for data entry. In the first box, coordinates for the analysis point are entered. In the second box, the optional radius in meters for the analysis area is specified. The third box requires the radius in meters of the area to be analyzed. Three buttons are available: the first for data submission, the second to access the results viewing page, and the third to log out. The third and final page allows for the visualization of the visual output of the tool. A single button, when pressed, generates an image in a few minutes, as shown in figure 4.5. This image adapts to the device in use, displaying a map with the conducted analysis.

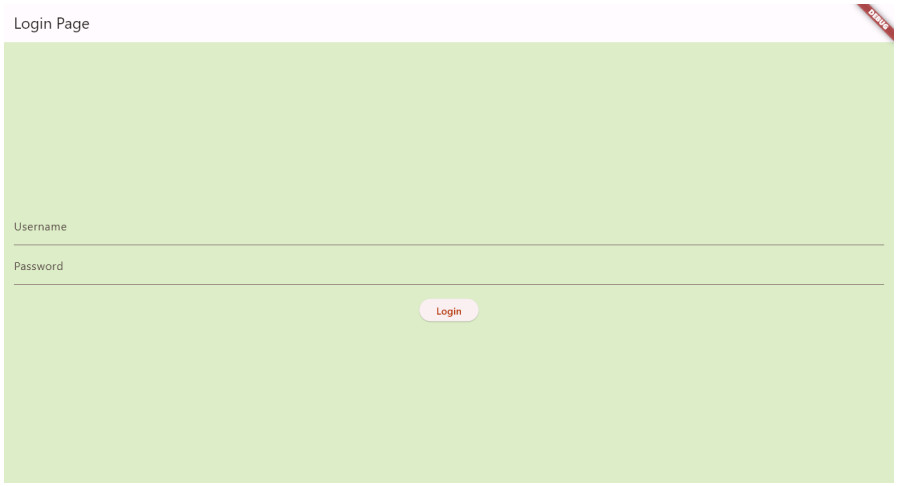


Figure 4.3: Login Page

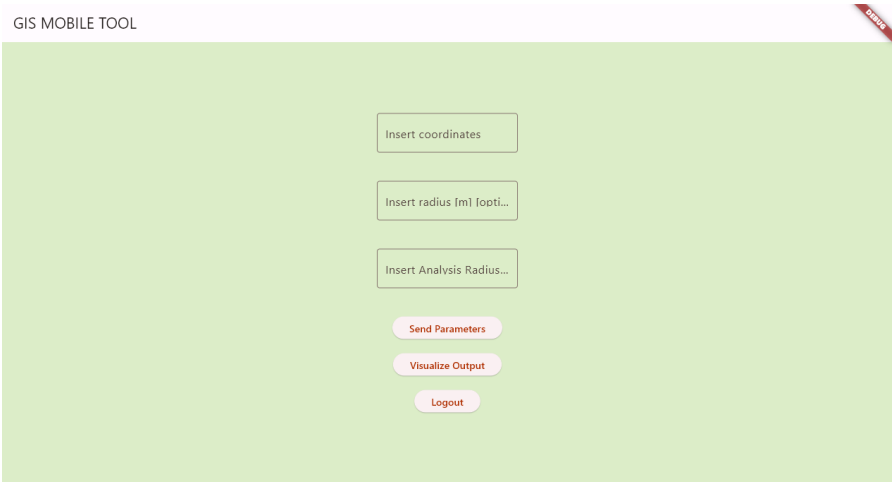


Figure 4.4: Data Page

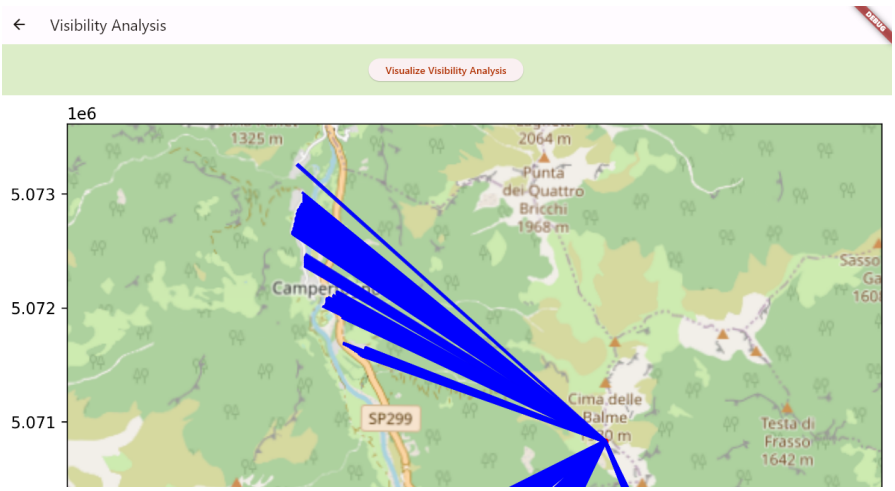


Figure 4.5: Analysis Page

4.2 Development

The GIS Mobile Tool is a comprehensive and sophisticated software that allows users to perform visibility analysis on mobile devices. The tool has been developed meticulously and with great attention to detail, using the powerful combination of Python programming language for the back-end and Flutter for the front-end. The database is built on SQLite, which offers high performance and reliability. Moreover, the tool is designed to be highly customizable, allowing users to tailor it to their specific needs. The algorithm development uses QGIS, ensuring the tool provides accurate and reliable results. QGIS allows data visualization using maps,

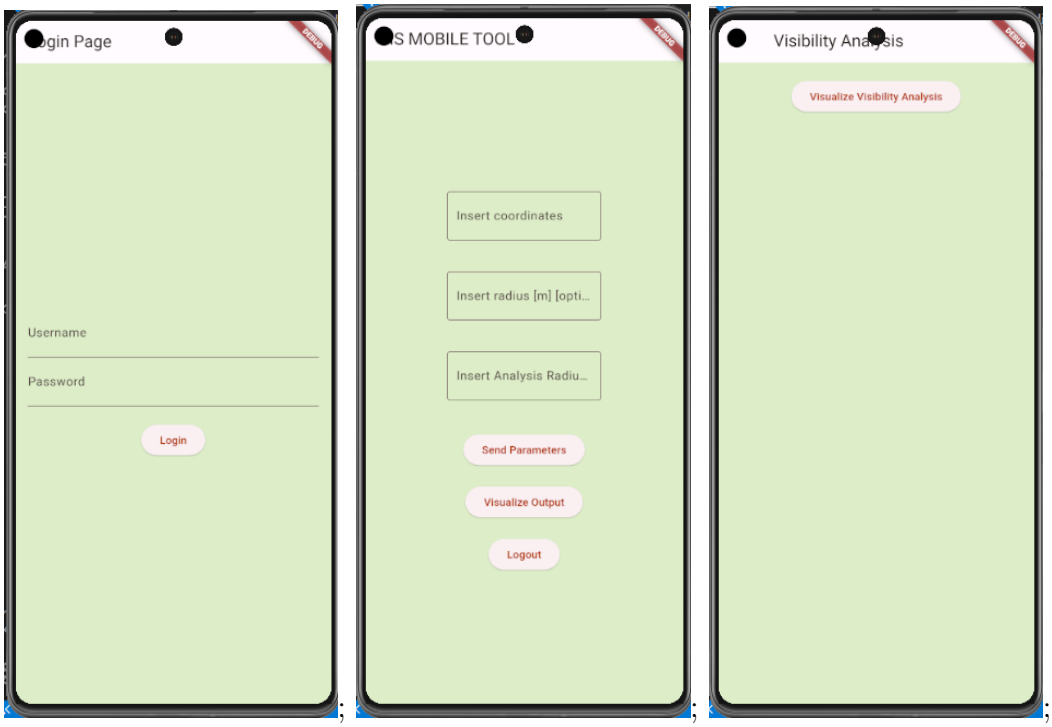


Figure 4.6: Application Pages on Smartphone

charts, and diagrams while customizing the presentation with various symbology choices. Flutter is an open-source framework by Google for building beautiful, natively compiled, multi-platform applications from a single codebase. [16] One of its key functionalities is enabling developers to build inviting and high-performance applications with a consistent user experience across multiple platforms. One of Flutter's standout features is its hot-reload capability, enabling developers to view changes made to the code instantly, stimulating a highly efficient development process. The framework provides a rich set of customizable widgets, ensuring flexibility and creativity in designing user interfaces. Furthermore, Flutter supports various plugins and packages, allowing developers to integrate various functionalities seamlessly. Its extensive community and growing ecosystem contribute to a wealth of resources and support for developers. With its cross-platform nature, Flutter simplifies the development process, reducing the time and effort required to create applications for different platforms. On the other hand, SQLite is an in-process library that implements a self-contained, serverless, zero-configuration, transactional SQL database engine. The code for SQLite is in the public domain and is thus free for use for any purpose, commercial or private. [17] SQLite stores data in a single, self-contained file, simplifying deployment and management. One of the critical features of SQLite is its support for SQL (Structured Query Language),

allowing developers to define, manipulate, and query relational databases using a standard and widely adopted syntax. SQLite is known for its efficiency, especially in terms of storage and resource utilization. It operates without a separate server process, reducing overhead and making it suitable for embedded systems and mobile applications. The database engine is also highly scalable and capable of handling large datasets and concurrent connections. Another noticeable aspect of SQLite is its cross-platform compatibility, supporting various operating systems, programming languages, and platforms. SQLite is a versatile and reliable database management system that excels in simplicity, efficiency, and cross-platform compatibility. It is ideal for various applications, from small scale embedded systems to large-scale software solutions.

4.2.1 Code development

A thorough understanding of how the algorithm operates and its final configuration is essential through a detailed examination of the source code. Subsequently, it will be possible to explore both the back-end and front-end code, paying particular attention to the implementation of the QGIS algorithm and the configuration of the bash file. Delving into the source code provides a clear overview of design choices, interactions between components, and key procedures that drive the proper functioning of the application. It also allows for a greater understanding of implementation decisions and optimizations made during the development of both the back-end and front-end. Figure 4.7 provides a detailed view of the tool development scheme, primarily divided between the front-end and back-end.

Front-End

Front-End code configuration can be observed in appendix B.1. The code is structured into several components, each serving a distinct purpose in the Flutter application. The initialization phase begins with the configuration and start of the Flutter app in the main() function. This involves essential imports and the definition of the main widget, 'MyApp' (B.1). The core of the application's logic lies in the 'MyAppState' class, an extension of 'ChangeNotifier'. This class handles global state management, incorporating methods for user authentication, logout, data submission to the back-end, and image retrieval. The seamless integration of authentication and API calls within 'MyAppState' streamlines the overall functionality (B.2). The primary user interface is represented by the 'MyHomePage' widget, which is conditionally displayed based on the user's authentication status. This widget features input fields for coordinates, radius, and analysis radius. Additionally, it provides buttons for submitting data, visualizing output, and logging out (B.3). The 'NewPage' widget specializes in displaying

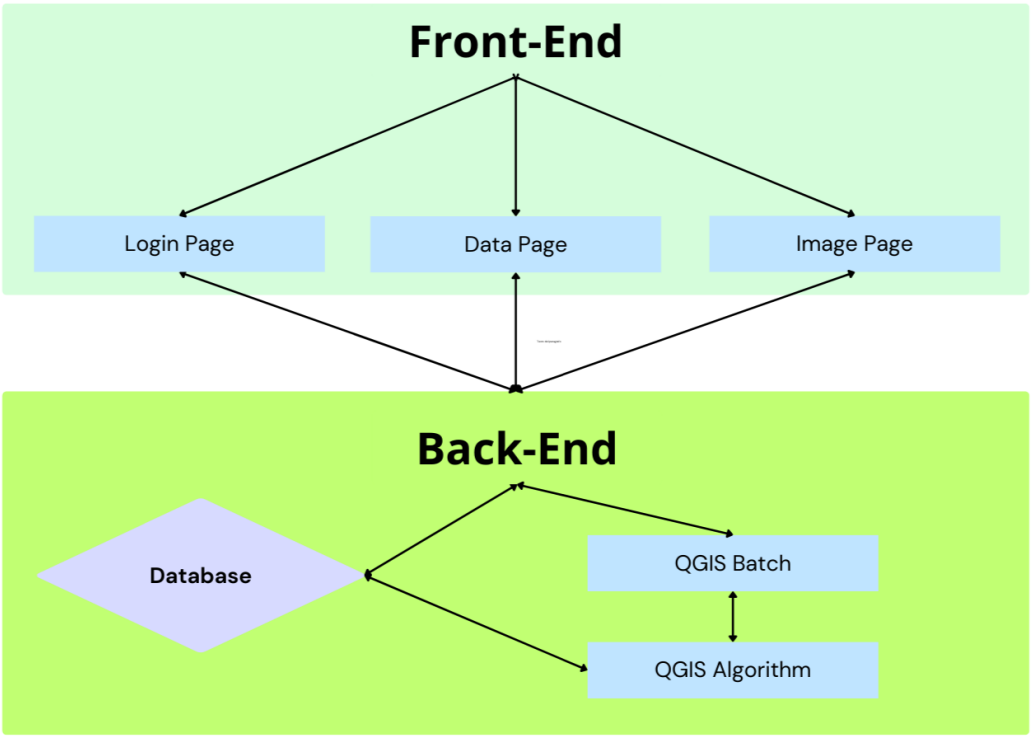


Figure 4.7: Development Scheme

output, particularly the visualization of visibility analysis. It facilitates the retrieval of images from the back-end, dynamically adjusting the display based on the availability of image data (B.4). The login functionality is encapsulated within the 'LoginPage' widget, which hosts the 'LoginForm' module. This module, a stateful widget, handles user input for the username and password, initiating the authentication process upon pressing the 'Login' button (B.5). The entire application comes to life through the Flutter app initialization in the main() function, culminating in the execution of runApp(MyApp()). This succinctly initiates the Flutter app, establishing a seamless flow of user interactions and backend communications (B.6).

Back-End

Back-End code configuration can be observed in appendix B.2. The Flask application begins its journey with the initialization of the app and the crucial configuration steps, such as enabling Cross-origin resource sharing (CORS) to handle cross-origin requests and establishing a secret key for secure session management (B.7). For

database operations, a connection function named connect db is introduced to seamlessly connect to the SQLite database, referred to as raffaele.db (B.8). User authentication and session management unfold through distinct endpoints. The /login endpoint meticulously verifies user credentials against the database, dynamically updating the session upon a successful login attempt. Conversely, the /logout endpoint adeptly clears the user's session, ensuring a secure logout process. The /check login endpoint serves as a sentinel, examining the current login status and providing valuable insights into the user's authentication state (B.9). The /send data endpoint becomes the gateway for data from the client, orchestrating an interaction with the SQLite database. This endpoint handles the task of updating existing records or introducing new data based on the coordinates provided by the user (B.10). The /get image endpoint executes a batch file to render geographical data using Geopandas and Matplotlib. The resulting image is carefully preserved and promptly delivered to the awaiting client, transforming abstract data into a tangible visual representation (B.11). Finally the app initializes to run on 0.0.0.0 at port 5000 in debug mode, providing a robust platform for testing and development (B.12).

QGIS Batch

QGIS Batch code configuration can be observed in appendix B.2.1. This script serves as a Windows batch file designed to configure the environment for an application relying on QGIS. The initial line, @echo off, suppresses the echoing of commands, contributing to a cleaner script execution without displaying each command. The script sets the 'OSGEO4W ROOT' variable to the installation directory of QGIS, and 'QGIS PREFIX' is defined as the path to the QGIS installation within the OSGEO4W root. Key system paths are then modified using the 'SET PATH' command. Directories containing QGIS and OSGEO4W binaries are appended to the system PATH variable, ensuring that the system recognizes the locations of QGIS and OSGEO4W binaries. The 'PYTHONPATH' environment variable is configured to include paths relevant to Python. This encompasses the QGIS Python directory, the Python directory within OSGEO4W, and QGIS Python plugins. The 'PYTHONHOME' variable is set to point to the location of the Python interpreter within the QGIS installation. Additionally, the 'QT PLUGIN PATH' variable is defined, indicating the path to the Qt5 plugins directory within the QGIS installation. The script proceeds to initiate the execution of a Python script named Test.py using the Python interpreter. This is achieved with the command python Test.py before exit (B.13).

QGIS Algorithm

QGIS Algorithm code configuration can be observed in appendix B.2.2. This initial section of the script is responsible for preparing the environment for QGIS. It adds the necessary paths for QGIS plugins, initializes the QGIS application, and incorporates the native algorithms provider. Furthermore, it introduces and configures a custom visibility provider, adding it to the processing registry. Finally, it initializes the Processing module, which is crucial for executing geospatial algorithms (B.14). The next script establishes a connection to an SQLite database named raffaele.db. It creates a cursor to interact with the database and executes a SELECT query on the 'DATA' table. The retrieved data is stored in the variable data from db for further processing. Finally, the database connection is closed to ensure proper resource management (B.15). The third script enters a loop to process each row of data retrieved from the database. Within this loop, it executes specific QGIS algorithms using the Processing framework. Notably, it generates points along a line and creates viewpoints for visibility analysis. The loop structure allows the script to perform these operations iteratively for each set of data from the database (B.16). The final section is responsible for exiting the QGIS application. It calls the 'exitQgis()' method to ensure proper cleanup and termination of the QGIS environment. This step is crucial for maintaining the stability of the application and freeing up resources associated with the QGIS session (B.17).

4.3 Results

The output produced by the GIS Mobile Tool is visualized through an image on the application, as depicted in figure 4.8. The output is displayed as an image on the application, and it is composed of a map derived from OpenStreetMap, with blue segments originating from the starting point and extending towards the surrounding roads. The tool conducts the analysis within a 10,000-meter radius, which allows for a comprehensive view of the area. The blue segments on the output image represent the distance from the starting point to the surrounding roads. This analysis provides valuable information to users, such as the distance between two points, the best route to take, and the accessibility of different areas. At the boundary of the output image, it is noticeable that the blue segments reach halfway towards the surrounding roads, which signifies a commendable result. This outcome highlights the accuracy of the algorithm used by the tool, which is crucial for users who rely on the output for decision-making purposes.

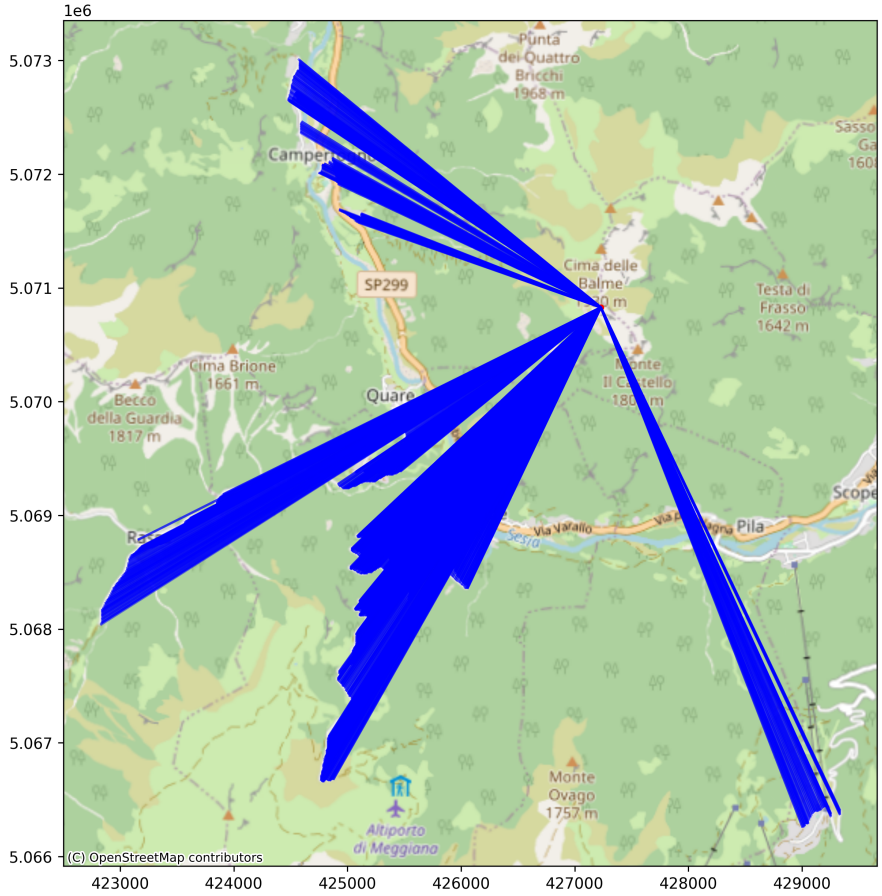


Figure 4.8: Visibility Analysis Result

4.3.1 Output Analysis

This section delves into the results obtained from the algorithm and elucidates the correlation between the input parameters and the algorithm's execution time. The algorithm's execution time is a crucial metric in achieving the set objectives and directly influences the planning phase. Inadequate time allocation during this phase can result in suboptimal outcomes. The first configuration under examination involves a single observation point as input, with area analysis radius variations. The graph 4.9 clearly illustrates the trend of execution time with six different input rays. The first analysis radius is 5000 meters, a vast range allowing a complete view. The plugin's execution time is just around 30 seconds, an excellent result in terms of planning. The most extensive analysis is conducted with a radius of 30000 meters, a comprehensive analysis that requires specific considerations for its use. This analysis requires around 8 minutes, which is a tolerable time compared to the

significant extension. Graph 4.9 shows that the times remain around two minute up to 15000 meters but spike at 20000 meters radius. This favorable outcome covers significant distances in a minute or less, and it is perfect for the requirements of that application. The second configuration under examination involves multiple observation points as input in a radius of 25 m, with area analysis radius variations. The graph 4.10 clearly illustrates the trend of execution time with five different input rays. The first analysis radius is 5000 meters, a vast range allowing a complete view. The plugin's execution time is just around 60 seconds, an excellent result in terms of planning. The most extensive analysis is conducted with a radius of 25000 meters, and it requires around 5 hours, which is a tolerable time compared to the significant extension. However, compared to other configurations, his time is higher. Graph 4.10 shows that the times remain around seven minutes up to 15000 meters but spike at 20000 meters radius. The outcome is coherent with the previous analysis and evidence that staying within 15000 meters for rapid request analysis is better. Examining these results, one can be satisfied with the achieved timelines. The GIS Mobile Tool is designed for on-the-fly situations without the need for pre-planning. The response times and modes are satisfactory to fulfill its intended purpose. The flexibility-oriented design allows the tool to adapt promptly to various circumstances, ensuring a timely and effective response when needed. The immediate and practical nature of the application proves particularly useful in contexts where rapid intervention is crucial, and advance planning might not be feasible.

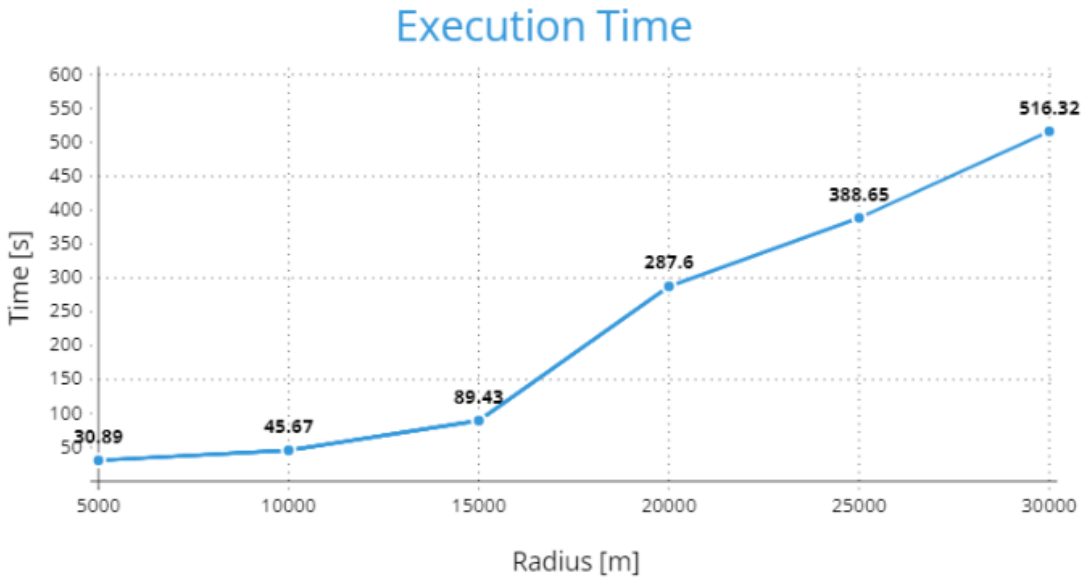


Figure 4.9: Execution time for single point configuration

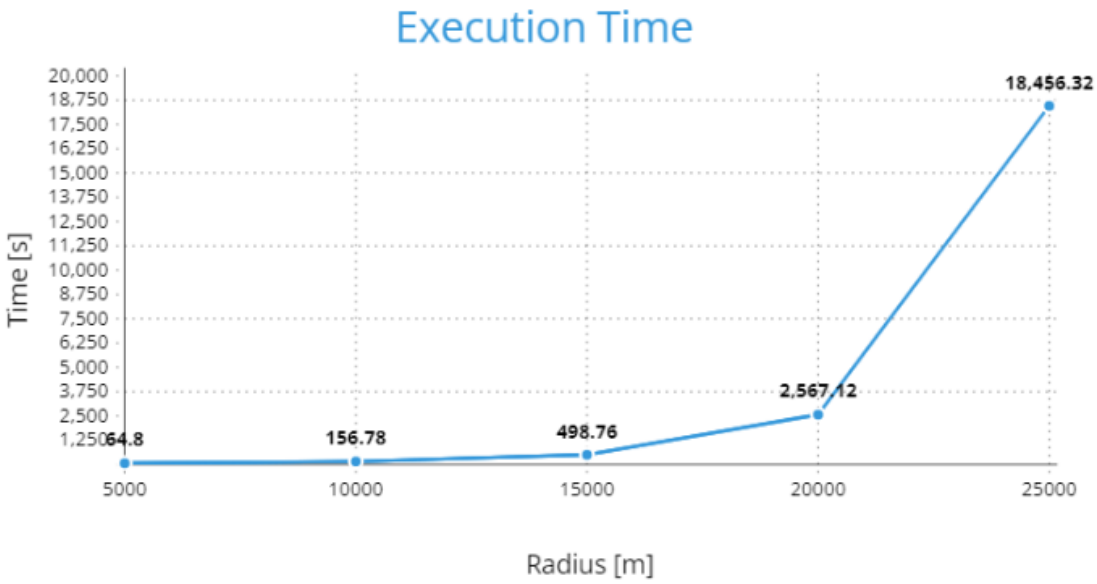


Figure 4.10: Execution time for 25 m area configuration

4.3.2 Limitations

Analyzing the results generated by the GIS Mobile tool is fundamental to delineating the limitations encountered. These limitations may manifest in various forms, ranging from input values that significantly impede algorithmic execution to constraints that necessitate the impracticability or redundancy of the algorithm. Exploring these limitations, it is possible to assert that they are very similar to the GIS Desk Tool ones. This approach is vital in creating solutions better suited to end-user needs. The first limitation is the analysis radius, which cannot exceed 30000 m as in the GIS Desk Tool case. This is usually sufficient for most applications, but it can be restrictive for assessments that require analysis beyond this range. In some cases, processing times can extend to several hours, affecting the tool's responsiveness. Another tool limitation involves the observation radius when multiple observation points are used. As the observation area radius increases, processing time increases significantly, especially when set above 25-30 m. This limitation aligns with the strategic nature of stakeouts, where the extension has a practical limit. The final limitation concerns data availability. Before utilizing the GIS Mobile Tool, it's necessary to prepare a database containing elevation models and street files. In certain instances, there may be a need to extract or generate this data for regions lacking it. This requirement introduces a preparatory step into the tool's workflow, limiting its adaptability to areas with pre-existing geospatial data. Another limitation arises from the tool's reliance on a single analysis, as

opposed to the two available in the GIS Desk Tool. Additionally, the analysis focuses solely on streets, excluding the broader surrounding area. This choice was primarily made to ensure the tool's responsiveness and speed. These limitations underscore the importance of considering practical constraints, processing times, and data availability when employing the GIS Mobile Tool in various contexts.

Chapter 5

Conclusion

This research analyzed the complexities and opportunities surrounding strategic military operations within the rapidly evolving technological landscape. The main goal was to address the increasing complexity of geopolitical dynamics by utilizing Geographic Information Systems (GIS) technology. The main challenges addressed were related to reducing planning timelines, the need for rapid adaptation to dangerous situations, and the ability to operate in conditions with limited connectivity. Military operations require careful planning and communication. In situations where time is limited, technology can enhance responsiveness and preparedness. Reliable communication is critical, and backup systems are necessary in case of connectivity issues. Safety is also paramount, and technology can ensure the safety of military personnel. The user needs to have an automatized area analysis in the planning phase, rapid responses on the field, working offline, conducting analyses starting from a point or an established area, and performing analyses without prior preparation. The research demonstrates that the use of two GIS tools, a desktop application for offline use and a mobile app for online scenarios, has significantly enhanced the military's capability to address operational challenges. By improving response times and conducting a detailed analysis of terrain visibility, the risks associated with mission failure have been drastically reduced. It is important to note that the adaptability and speed of planning have also been greatly enhanced, allowing for greater flexibility in military operations. The use of GIS tools has the potential to revolutionize planning by saving precious time and resources. This innovation helps to address several challenges, including adapting to dangerous situations and proactively preventing issues in secure environments. The impact of these tools extends beyond just operational efficiency. They can represent a change point in planning and responsiveness on the field, allowing for a strategic advantage in every situation. It is important to say that these two tools operate in different conditions, yet exhibit very similar response times, enabling their combined use in missions to cover a broader range of scenarios. It is

crucial to emphasize that the tools adapt to various available map data, whether open, already available, or created. Looking ahead, there are many opportunities for further development. Continuously refining these applications and making them more accessible can enhance operational efficiency. Implementing advanced algorithms with collaborations with other disciplines have the potential to yield significant advancements. An important enhancement could be the integration of a tool capable of autonomously recognizing roads around a designated point. This capability would further reduce planning timelines and enhance the overall efficiency of the tools. A further significant development could be the adaptation of the GIS Mobile Tool also for Apple devices with the iOS operating system. At present, the use of the tool on Android systems has an obvious limitation, as those who do not have a device of this system are excluded from its use. The utilization of GIS tools signifies a substantial leap forward, affording military leaders a clearer perspective and a more rapid response to the continually evolving operational landscape. This research evidence the significance of using advanced technologies to enhance the safety of individuals, potentially making the workplace more comfortable and even saving lives.

Appendix A

GIS Desk Tool

A.1 Plugin code development

A.1.1 Library import

```
1
2 __author__ = 'Raffaele Pezone'
3 __date__ = '2023-10-01'
4 __copyright__ = '(C) 2023 by Raffaele Pezone'
5
6 # This will get replaced with a git SHA1 when you do a git archive
7
8 __revision__ = '$Format:%H$'
9
10 from qgis.PyQt.QtCore import QCoreApplication
11 from qgis.core import (QgsProcessing,
12                        QgsFeatureSink,
13                        QgsProject,
14                        QgsRasterLayer,
15                        QgsProcessingAlgorithm,
16                        QgsProcessingParameterNumber,
17                        QgsProcessingMultiStepFeedback,
18                        QgsProcessingParameterRasterLayer,
19                        QgsProcessingParameterPoint,
20                        QgsProcessingParameterMapLayer,
21                        QgsProcessingParameterFeatureSource,
22                        QgsProcessingParameterFeatureSink,
23                        QgsProcessingParameterRasterDestination)
24 import processing
```

A.1.2 Class VisibilityAnalysisToolAlgorithm

```
1
2 class VisibilityAnalysisToolAlgorithm(QgsProcessingAlgorithm):
3
4     OUTPUT = 'OUTPUT'
5     INPUT = 'INPUT'
```

Parameters initialization method

```
1
2     def initAlgorithm(self, config):
3         """
4         Here we define the inputs and output of the algorithm, along
5         with some other properties.
6         """
7         self.addParameter(QgsProcessingParameterRasterLayer('
    dtm_piemonte', 'DTM or DSM', defaultValue=None))
8         self.addParameter(QgsProcessingParameterPoint('punto_di_oss2'
    , 'Observation Point Coordinates', defaultValue=None))
9         self.addParameter(QgsProcessingParameterNumber('raggio_pol',
    'Polygon Radius [m]', type=QgsProcessingParameterNumber.Double,
    minValue=0, maxValue=100, defaultValue=0, optional=True))
10        self.addParameter(QgsProcessingParameterNumber('
    raggio_analisi_1', 'Analysis Radius 1 [m]', type=
    QgsProcessingParameterNumber.Double, minValue=0, maxValue=30000,
    defaultValue=10000))
11        self.addParameter(QgsProcessingParameterNumber('
    raggio_analisi_2', 'Analysis Radius 2 [m]', type=
    QgsProcessingParameterNumber.Double, minValue=0, maxValue=30000,
    defaultValue=0, optional=True))
12        self.addParameter(QgsProcessingParameterNumber('altezza oss',
     'Observer Height [m]', type=QgsProcessingParameterNumber.Double,
    defaultValue=1.6))
13        self.addParameter(QgsProcessingParameterNumber('altezza targ'
    , 'Target Height [m]', type=QgsProcessingParameterNumber.Double,
    defaultValue=0))
14        self.addParameter(QgsProcessingParameterMapLayer('strada', '
    Streets', defaultValue=None, types=[QgsProcessing.TypeVectorLine])
    )
15        self.addParameter(QgsProcessingParameterFeatureSink('Strade',
     'Streets', type=QgsProcessing.TypeVectorAnyGeometry,
    createByDefault=True, defaultValue=None))
```

```
16          self.addParameter(QgsProcessingParameterFeatureSink('
    FinalResult2', 'Final Result Analysis 2', type=QgsProcessing.
    TypeVectorAnyGeometry, createByDefault=True, defaultValue=None))
17          self.addParameter(QgsProcessingParameterFeatureSink('
    FinalResult', 'Final Result Analysis 1', type=QgsProcessing.
    TypeVectorAnyGeometry, createByDefault=True, defaultValue=None))
18          self.addParameter(QgsProcessingParameterRasterDestination('
    Analisi', 'Area Analysis', createByDefault=True, defaultValue=None
    ))
19          self.addParameter(QgsProcessingParameterFeatureSink('Si', '
    Observation Point', type=QgsProcessing.TypeVectorPoint,
    createByDefault=True, defaultValue=None))
```

Process algorithm method

```
1
2       def processAlgorithm(self, parameters, context, model_feedback):
3           """
4           Here is where the processing itself takes place.
5           """
6           # Use a multi-step feedback, so that individual child
    algorithm progress reports are adjusted for the
7           # overall progress through the model
8           feedback = QgsProcessingMultiStepFeedback(5, model_feedback)
9           results = {}
10          outputs = {}
```

Listing A.1: Create points pixel along the line

```
1
2           # Create points pixel along the line
3           alg_params = {
4               'INPUT_RASTER': parameters['dtm_piemonte'],
5               'INPUT_VECTOR': parameters['strada'],
6               'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
7           }
8           outputs['CreaPuntiCentroidiDelPixelLungoLaLinea'] =
    processing.run('qgis:generatepointspixelcentroidsalongline',
    alg_params, context=context, feedback=feedback, is_child_algorithm
    =True)
9
10          feedback.setCurrentStep(1)
11          if feedback.isCanceled():
12              return {}
```

Listing A.2: Create observation point

```
1
2         if parameters['punto_di_oss2'] != None:
3
4                 # Observation point
5                 alg_params = {
6                     'INPUT': parameters['punto_di_oss2'],
7                     'OUTPUT': parameters['Si']
8                 }
9                 outputs['PuntoDiOsservazione'] = processing.run('
    native:pointtolayer', alg_params, context=context, feedback=
    feedback, is_child_algorithm=True)
10                results['Si'] = outputs['PuntoDiOsservazione']['
    OUTPUT']
11
12                inp = outputs['PuntoDiOsservazione']['OUTPUT']
13
14                feedback.setCurrentStep(3)
15                if feedback.isCanceled():
16                    return {}
```

Listing A.3: Create optional observation area

```
1
2         if parameters['raggio_pol'] != 0:
3
4                 # Buffer
5                 alg_params = {
6                     'DISSOLVE': False,
7                     'DISTANCE': parameters['raggio_pol'],
8                     'END_CAP_STYLE': 0,  # Arrotondato
9                     'INPUT': outputs['PuntoDiOsservazione']['
    OUTPUT'],
10                    'JOIN_STYLE': 0,  # Arrotondato
11                    'MITER_LIMIT': 2,
12                    'SEGMENTS': 5,
13                    'SEPARATE_DISJOINT': False,
14                    'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
15                }
16                outputs['Buffer'] = processing.run('native:buffer
    ', alg_params, context=context, feedback=feedback,
    is_child_algorithm=True)
```

Listing A.4: Create point inside the optional observation area

```
1
2                 # Generate point inside the polygon
3                 alg_params = {
4                     'INPUT_RASTER': parameters['dtm_piemonte'],
```

```
                    'INPUT_VECTOR': outputs['Buffer']['OUTPUT'],
                    'OUTPUT': parameters['Si']
                }
                outputs['GeneraPuntiCentroidiDeiPixelDentroPoligoni'] = processing.run('native:generatepointspixelcentroidsinsidepolygons', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
                results['Si'] = outputs['GeneraPuntiCentroidiDeiPixelDentroPoligoni']['OUTPUT']

                inp = outputs['GeneraPuntiCentroidiDeiPixelDentroPoligoni']['OUTPUT']

                feedback.setCurrentStep(1)
                if feedback.isCanceled():
                    return {}
```

Listing A.5: Create observation viewpoint

```

        # PUNTO
        alg_params = {
            'ANGLE_DOWN_FIELD': '',
            'ANGLE_UP_FIELD': '',
            'AZIM_1_FIELD': '',
            'AZIM_2_FIELD': '',
            'DEM': parameters['dtm_piemonte'],
            'OBSERVER_ID': '',
            'OBSERVER_POINTS': inp,
            'OBS_HEIGHT': parameters['altezza oss'],
            'OBS_HEIGHT_FIELD': '',
            'RADIUS': parameters['raggio_analisi_1'],
            'RADIUS_FIELD': '',
            'RADIUS_IN_FIELD': '',
            'TARGET_HEIGHT': parameters['altezza targ'],
            'TARGET_HEIGHT_FIELD': '',
            'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
        }
        outputs['Punto'] = processing.run('visibility:createviewpoints', alg_params, context=context, feedback=feedback, is_child_algorithm=True)

        feedback.setCurrentStep(2)
        if feedback.isCanceled():
            return {}
```

Listing A.6: Create streets viewpoint

```

```

```
        # STR
        alg_params = {
            'ANGLE_DOWN_FIELD': '',
            'ANGLE_UP_FIELD': '',
            'AZIM_1_FIELD': '',
            'AZIM_2_FIELD': '',
            'DEM': parameters['dtm_piemonte'],
            'OBSERVER_ID': '',
            'OBSERVER_POINTS': outputs['CreaPuntiCentroidiDelPixelLungoLaLinea']['OUTPUT'],
            'OBS_HEIGHT': parameters['altezza oss'],
            'OBS_HEIGHT_FIELD': '',
            'RADIUS': parameters['raggio_analisi_1'],
            'RADIUS_FIELD': '',
            'RADIUS_IN_FIELD': '',
            'TARGET_HEIGHT': parameters['altezza targ'],
            'TARGET_HEIGHT_FIELD': '',
            'OUTPUT': parameters['Strade']
        }
        outputs['Str'] = processing.run('visibility:createviewpoints', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
        results['Strade'] = outputs['Str']['OUTPUT']

        feedback.setCurrentStep(3)
        if feedback.isCanceled():
            return {}
```

Listing A.7: Create viewshed

```

        # Viewshed
        alg_params = {
            'ANALYSIS_TYPE': 0,  # Binary viewshed
            'DEM': parameters['dtm_piemonte'],
            'OBSERVER_POINTS': outputs['Punto']['OUTPUT'],
            'OPERATOR': 1,  # Addition
            'REFRACTION': 0.13,
            'USE_CURVATURE': True,
            'OUTPUT': parameters['Analisi']
        }
        outputs['Viewshed'] = processing.run('visibility:viewshed', alg_params, context=context, feedback=feedback, is_child_algorithm=True)
        results['Analisi'] = outputs['Viewshed']['OUTPUT']

        feedback.setCurrentStep(4)
        if feedback.isCanceled():
            return {}
```

Listing A.8: Create intervisibility network

```
1
2         # Intervisibility network
3         alg_params = {
4             'DEM': parameters['dtm_piemonte'],
5             'OBSERVER_POINTS': outputs['Punto']['OUTPUT'],
6             'REFRACTION': 0.13,
7             'TARGET_POINTS': outputs['Str']['OUTPUT'],
8             'USE_CURVATURE': True,
9             'WRITE_NEGATIVE': False,
10            'OUTPUT': parameters['FinalResult']
11        }
12        outputs['IntervisibilityNetwork'] = processing.run('
    visibility:intervisibility', alg_params, context=context, feedback
    =feedback, is_child_algorithm=True)
13        results['FinalResult'] = outputs['IntervisibilityNetwork']['
    OUTPUT']
```

Listing A.9: Create observation viewpoint 2

```
1
2         if parameters['raggio_analisi_2'] != 0:
3                 # PUNTO 2
4                 alg_params = {
5                     'ANGLE_DOWN_FIELD': '',
6                     'ANGLE_UP_FIELD': '',
7                     'AZIM_1_FIELD': '',
8                     'AZIM_2_FIELD': '',
9                     'DEM': parameters['dtm_piemonte'],
10                    'OBSERVER_ID': '',
11                    'OBSERVER_POINTS': inp,
12                    'OBS_HEIGHT': parameters['altezza oss'],
13                    'OBS_HEIGHT_FIELD': '',
14                    'RADIUS': parameters['raggio_analisi_2'],
15                    'RADIUS_FIELD': '',
16                    'RADIUS_IN_FIELD': '',
17                    'TARGET_HEIGHT': parameters['altezza targ'],
18                    'TARGET_HEIGHT_FIELD': '',
19                    'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
20                }
21                outputs['Punto2'] = processing.run('visibility:
    createviewpoints', alg_params, context=context, feedback=feedback,
     is_child_algorithm=True)
22
23                feedback.setCurrentStep(2)
24                if feedback.isCanceled():
25                    return {}
```

Listing A.10: Create streets viewpoint

```
1
2                   # STR 2
3                   alg_params = {
4                       'ANGLE_DOWN_FIELD': '',
5                       'ANGLE_UP_FIELD': '',
6                       'AZIM_1_FIELD': '',
7                       'AZIM_2_FIELD': '',
8                       'DEM': parameters['dtm_piemonte'],
9                       'OBSERVER_ID': '',
10                      'OBSERVER_POINTS': outputs['
    CreaPuntiCentroidiDelPixelLungoLaLinea']['OUTPUT'],
11                      'OBS_HEIGHT': parameters['altezza oss'],
12                      'OBS_HEIGHT_FIELD': '',
13                      'RADIUS': parameters['raggio_analisi_2'],
14                      'RADIUS_FIELD': '',
15                      'RADIUS_IN_FIELD': '',
16                      'TARGET_HEIGHT': parameters['altezza targ'],
17                      'TARGET_HEIGHT_FIELD': '',
18                      'OUTPUT': QgsProcessing.TEMPORARY_OUTPUT
19                  }
20                  outputs['Str2'] = processing.run('visibility:
    createviewpoints', alg_params, context=context, feedback=feedback,
     is_child_algorithm=True)
21
22
23                  feedback.setCurrentStep(3)
24                  if feedback.isCanceled():
25                      return {}
```

Listing A.11: Create intervisibility network 2

```
1
2                   # Intervisibility network2
3                   alg_params = {
4                       'DEM': parameters['dtm_piemonte'],
5                       'OBSERVER_POINTS': outputs['Punto2']['OUTPUT'],
6                       'REFRACTION': 0.13,
7                       'TARGET_POINTS': outputs['Str2']['OUTPUT'],
8                       'USE_CURVATURE': True,
9                       'WRITE_NEGATIVE': False,
10                      'OUTPUT': parameters['FinalResult2']
11                  }
12                  outputs['IntervisibilityNetwork2'] = processing.run('
    visibility:intervisibility', alg_params, context=context, feedback
    =feedback, is_child_algorithm=True)
13                  results['FinalResult2'] = outputs['
    IntervisibilityNetwork2']['OUTPUT']
14
```

```
15
16          return results
```

A.1.3 Name definition method

```
1
2       def name(self):
3           """
4           Returns the algorithm name, used for identifying the
    algorithm. This
5           string should be fixed for the algorithm, and must not be
    localised.
6           The name should be unique within each provider. Names should
    contain
7           lowercase alphanumeric characters only and no spaces or other
8           formatting characters.
9           """
10          return 'Visibility Analysis Tool'
```

A.1.4 DisplayName definition method

```
1
2       def displayName(self):
3           """
4           Returns the translated algorithm name, which should be used
    for any
5           user-visible display of the algorithm name.
6           """
7           return self.tr(self.name())
```

A.1.5 Group definition method

```
1
2       def group(self):
3           """
4           Returns the name of the group this algorithm belongs to. This
     string
5           should be localised.
6           """
```

```
7         return self.tr(self.groupId())
```

A.1.6 GroupID definition method

```
1
2     def groupId(self):
3         """
4         Returns the unique ID of the group this algorithm belongs to. This
5         string should be fixed for the algorithm, and must not be localised.
6         The group id should be unique within each provider. Group id should
7         contain lowercase alphanumeric characters only and no spaces or other
8         formatting characters.
9         """
10        return ''
```

A.1.7 Try definition method

```
1
2     def tr(self, string):
3         return QCoreApplication.translate('Processing', string)
```

A.1.8 Create instance definition method

```
1
2     def createInstance(self):
3         return VisibilityAnalysisToolAlgorithm()
```

Appendix B

GIS Mobile Tool

B.1 Front-End

Listing B.1: MyApp definition

```
1
2 import 'dart:convert';
3 import 'dart:typed_data';
4 import 'package:http/http.dart' as http;
5 import 'package:flutter/material.dart';
6 import 'package:provider/provider.dart';
7
8 void main() {
9   runApp(MyApp());
10 }
```

Listing B.2: MyAppState

```
1 class MyApp extends StatelessWidget {
2   @override
3   Widget build(BuildContext context) {
4     return ChangeNotifierProvider(
5       create: (context) => MyAppState(),
6       child: MaterialApp(
7         title: 'Namer App',
8         theme: ThemeData(
9           useMaterial3: true,
10          colorScheme: ColorScheme.fromSeed(seedColor: Colors.
    deepOrange),
11          scaffoldBackgroundColor: Colors.lightGreen[100],
12        ),
13        home: MyHomePage(),
14      ),
15    );
```

```
16    }
17  }
18
19  class MyAppState extends ChangeNotifier {
20    bool _isLoggedIn = false;
21
22    bool get isLoggedIn => _isLoggedIn;
23
24    final TextEditingController _coordinateController =
       TextEditingController();
25    final TextEditingController _radiusController =
       TextEditingController();
26    final TextEditingController _analysisRadiusController =
27        TextEditingController();
28
29    void authenticate(String username, String password) async {
30      final response = await http.post(
31        Uri.parse('http://192.168.1.227:5000/login'),
32        headers: {
33          'Content-Type': 'application/json',
34          'X-Requested-With': 'XMLHttpRequest',
35        },
36        body: jsonEncode({
37          'username': username,
38          'password': password,
39        }),
40      );
41
42      if (response.statusCode == 200) {
43        _isLoggedIn = true;
44        notifyListeners();
45      } else {
46        print('Errore di autenticazione');
47      }
48    }
49
50    void logout() async {
51      final response = await http.post(
52        Uri.parse('http://192.168.1.227:5000/logout'),
53      );
54
55      if (response.statusCode == 200) {
56        _isLoggedIn = false;
57        _coordinateController.text = '';
58        _radiusController.text = '';
59        _analysisRadiusController.text = '';
60        notifyListeners();
61      } else {
62        print('Errore di logout');
```

```
63      }
64    }
65
66    void sendDataToBackend(
67        String coordinates, String radius, String analysisRadius) async
        {
68      Map<String, dynamic> dataMap = {
69        'Coordinates': coordinates,
70        'Radius': radius,
71        'AnalysisRadius': analysisRadius,
72      };
73
74      final response = await http.post(
75        Uri.parse('http://192.168.1.227:5000/send_data'),
76        headers: {
77          'Content-Type': 'application/json',
78        },
79        body: jsonEncode(dataMap),
80      );
81
82      if (response.statusCode == 200) {
83        print('Risposta dal backend:');
84      } else {
85        print('Errore di invio dati al backend:');
86      }
87    }
88
89    void getImage(BuildContext context) async {
90      final response = await http.get(
91        Uri.parse('http://192.168.1.227:5000/get_image'),
92        headers: {
93          'Connection': 'keep-alive', // Adding Keep-Alive header
94        },
95      );
96
97      if (response.statusCode == 200) {
98        print('Lunghezza immagine: ${response.bodyBytes.length}');
99        Navigator.push(
100         context,
101         MaterialPageRoute(
102           builder: (context) => NewPage(imageBytes: response.
        bodyBytes),
103         ),
104       );
105     } else {
106       print('Errore durante il recupero dell\'immagine');
107     }
108   }
109 }
```

Listing B.3: MyHomePage

```
1
2  class MyHomePage extends StatelessWidget {
3    @override
4    Widget build(BuildContext context) {
5      var appState = Provider.of<MyAppState>(context);
6
7      if (appState.isLoggedIn) {
8        return Scaffold(
9          appBar: AppBar(
10           title: Text('GIS MOBILE TOOL'),
11         ),
12         body: Center(
13           child: Column(
14             mainAxisAlignment: MainAxisAlignment.center,
15             children: [
16               Container(
17                 width: 200,
18                 child: TextField(
19                   controller: appState._coordinateController,
20                   decoration: InputDecoration(
21                     labelText: 'Insert coordinates',
22                     border: OutlineInputBorder(),
23                   ),
24                 ),
25               ),
26               SizedBox(height: 40),
27               Container(
28                 width: 200,
29                 child: TextField(
30                   controller: appState._radiusController,
31                   decoration: InputDecoration(
32                     labelText: 'Insert radius [m] [optional]',
33                     border: OutlineInputBorder(),
34                   ),
35                 ),
36               ),
37               SizedBox(height: 40),
38               Container(
39                 width: 200,
40                 child: TextField(
41                   controller: appState._analysisRadiusController,
42                   decoration: InputDecoration(
43                     labelText: 'Insert Analysis Radius [m]',
44                     border: OutlineInputBorder(),
45                   ),
46                 ),
47               ),
48               SizedBox(height: 40),
```

```
49                ElevatedButton(
50                  onPressed: () {
51                    String coordinates = appState._coordinateController
    .text;
52                    String radius = appState._radiusController.text;
53                    String analysisRadius =
54                        appState._analysisRadiusController.text;
55                    appState.sendDataToBackend(
56                        coordinates, radius, analysisRadius);
57                    print('button pressed!');
58                  },
59                  child: Text('Send Parameters'),
60                ),
61                SizedBox(height: 20),
62                ElevatedButton(
63                  onPressed: () {
64                    Navigator.push(
65                      context,
66                      MaterialPageRoute(
67                          builder: (context) => NewPage(imageBytes:
    null)),
68                    );
69                  },
70                  child: Text('Visualize Output'),
71                ),
72                SizedBox(height: 20),
73                ElevatedButton(
74                  onPressed: () {
75                    appState.logout();
76                  },
77                  child: Text('Logout'),
78                ),
79              ],
80            ),
81          ),
82        );
83      } else {
84        return LoginPage();
85      }
86    }
87  }
```

Listing B.4: NewPage

```
1
2  class NewPage extends StatelessWidget {
3    final List<int>? imageBytes;
4
5    NewPage({required this.imageBytes});
```

```
6
7   @override
8   Widget build(BuildContext context) {
9     var appState = Provider.of<MyAppState>(context);
10
11    return Scaffold(
12      appBar: AppBar(
13        title: Text('Visibility Analysis'),
14      ),
15      body: SingleChildScrollView(
16        child: Center(
17          child: Column(
18            mainAxisAlignment: MainAxisAlignment.center,
19            children: [
20              SizedBox(height: 20),
21              ElevatedButton(
22                onPressed: () {
23                  appState.getImage(context);
24                },
25                child: Text('Visualize Visibility Analysis'),
26              ),
27              SizedBox(height: 20),
28              if (imageBytes != null)
29                Container(
30                  width: double.infinity, // Use the full width
    available
31                  child: Image.memory(
32                    Uint8List.fromList(imageBytes!),
33                    fit: BoxFit.contain, // Choose the appropriate
    fit
34                  ),
35                ),
36            ],
37          ),
38        ),
39      ),
40    );
41  }
42 }
```

Listing B.5: LoginPage

```
1
2 class LoginPage extends StatelessWidget {
3   @override
4   Widget build(BuildContext context) {
5     return Scaffold(
6       appBar: AppBar(
7         title: Text('Login Page'),
```

```
8         ),
9         body: Padding(
10          padding: const EdgeInsets.all(16.0),
11          child: LoginForm(),
12        ),
13      );
14    }
15  }
```

Listing B.6: LoginForm

```
1
2  class LoginForm extends StatefulWidget {
3    @override
4    _LoginFormState createState() => _LoginFormState();
5  }
6
7  class _LoginFormState extends State<LoginForm> {
8    final TextEditingController _usernameController =
       TextEditingController();
9    final TextEditingController _passwordController =
       TextEditingController();
10
11   @override
12   Widget build(BuildContext context) {
13     return Column(
14       crossAxisAlignment: CrossAxisAlignment.center,
15       mainAxisAlignment: MainAxisAlignment.center,
16       children: [
17         TextField(
18           controller: _usernameController,
19           decoration: InputDecoration(labelText: 'Username'),
20         ),
21         TextField(
22           controller: _passwordController,
23           obscureText: true,
24           decoration: InputDecoration(labelText: 'Password'),
25         ),
26         SizedBox(height: 20),
27         ElevatedButton(
28           onPressed: () {
29             var appState = Provider.of<MyAppState>(context, listen:
       false);
30             appState.authenticate(
31               _usernameController.text,
32               _passwordController.text,
33             );
34           },
35           child: Text('Login'),
```

```
            ),
        ],
    );
  }
}
```

B.2 Back-End

Listing B.7: Configuration

```

from flask import Flask, request, jsonify, send_file, session
from flask_cors import CORS
import sqlite3
import json
import geopandas as gpd
import matplotlib.pyplot as plt
import contextily as ctx
import subprocess

app = Flask(__name__)
CORS(app)

app.secret_key = 'your_secret_key'
```

Listing B.8: Database Connection

```

# Database Connection
def connect_db():
    return sqlite3.connect('raffaele.db')
```

Listing B.9: Authentication

```

@app.route('/login', methods=['POST'])
def login():
    data = request.get_json()

    username = data.get('username')
    password = data.get('password')

    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute('SELECT * FROM USER WHERE username=? AND password
    =?', (username, password))
    user = cursor.fetchone()
```

```
13      conn.close()
14
15      if user:
16          session['username'] = username
17          return jsonify({'message': 'Login successo'}), 200
18      else:
19          return jsonify({'message': 'Credenziali non valide'}), 401
20
21  # Logout
22  @app.route('/logout', methods=['POST'])
23  def logout():
24      session.pop('username', None)
25      return jsonify({'message': 'Logout successo'}), 200
26
27  # Login Check
28  @app.route('/check_login', methods=['GET'])
29  def check_login():
30      if 'username' in session:
31          return jsonify({'logged_in': True, 'username': session['
        username']}), 200
32      else:
33          return jsonify({'logged_in': False}), 200
```

Listing B.10: Data Send

```
1
2   # Endpoint
3   @app.route('/send_data', methods=['POST'])
4   def send_data():
5       data = request.get_json()
6       print('Dati ricevuti:', data)
7
8       coordinates = data["Coordinates"]
9       radius = data["Radius"]
10      radius1 = data["AnalysisRadius"]
11
12      conn = connect_db()
13      cursor = conn.cursor()
14      cursor.execute('SELECT * FROM DATA WHERE Coordinates=?', (
        coordinates,))
15      existing_data = cursor.fetchone()
16
17      if existing_data:
18          cursor.execute('UPDATE DATA SET Radius=?, AnalysisRadius=?
        WHERE Coordinates=?', (radius, radius1, coordinates))
19
20      else:
21          cursor.execute('INSERT INTO DATA (Coordinates, Radius,
        AnalysisRadius) VALUES (?, ?, ?)', (coordinates, radius, radius1))
```

```
22
23      conn.commit()
24      conn.close()
25
26      return jsonify({'message': 'Dati ricevuti con successo e salvati
        nel database'}), 200
```

Listing B.11: Get Image

```
1
2  @app.route('/get_image', methods=['GET'])
3  def get_image():
4
5      percorso_file_batch = r"C:/Users/raffa/Desktop/APPLICAZIONE/
       QGISenv.bat"
6
7      result = subprocess.run([percorso_file_batch], shell=True)
8
9      if result.returncode == 0:
10         print("Il file batch è stato eseguito con successo.")
11     else:
12         print(f"Errore durante l'esecuzione del file batch. Codice di
       ritorno: {result.returncode}")
13
14     file_path_poligono = "C:/Users/raffa/Desktop/INTER.gpkg"
15     file_path_punto = "C:/Users/raffa/Desktop/PUNTO.gpkg"
16
17     gdf_poligono = gpd.read_file(file_path_poligono)
18     gdf_punto = gpd.read_file(file_path_punto)
19
20     fig, ax = plt.subplots(figsize=(10, 10))
21
22     gdf_punto.plot(ax=ax, color='red', markersize=10)
23
24     gdf_poligono.plot(ax=ax, color='blue', alpha=0.5)
25
26     ctx.add_basemap(ax, crs=gdf_poligono.crs.to_string(), source=ctx.
       providers.OpenStreetMap.Mapnik)
27
28     image_path = 'C:/Users/raffa/Desktop/foto1.png'
29     fig.savefig(image_path, bbox_inches='tight', dpi=300)
30
31     plt.close(fig)
32
33     return send_file(image_path, mimetype='image/png')
```

Listing B.12: Flask App Launch

```
1
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

B.2.1 Batch File

Listing B.13: QGIS Environment Set

```
@echo off
REM This is a bootstrap script for Windows to set up the environment
    for an application that depends on QGIS.

SET OSGEO4W_ROOT=C:\Program Files\QGIS 3.32.0
SET QGIS_PREFIX=%OSGEO4W_ROOT%\apps\qgis
SET PATH=%QGIS_PREFIX%\bin;%OSGWO4W_ROOT%\bin;%PATH%
SET PYTHONPATH=%QGIS_PREFIX%\python;%OSEO4W_ROOT%\apps\Python39;%
    QGIS_PREFIX%\python\plugins;%PYTHONPATH%
SET PYTHONHOME=%OSGEO4W_ROOT%\apps\Python39
SET QT_PLUGIN_PATH = %OSGEO4W_ROOT%\apps\Qt5\plugins


REM Launch python job

python Test.py

REM Esci dallo script
exit
```

B.2.2 QGIS Algorithm

Listing B.14: Inizializing

```
from qgis.core import QgsApplication, QgsProcessing
from qgis.analysis import QgsNativeAlgorithms

import sqlite3
import sys
sys.path.append('C:/Program Files/QGIS 3.32.0/apps/qgis/python/
    plugins')

qgs = QgsApplication([], True)

# Load providers
qgs.initQgis()
QgsApplication.processingRegistry().addProvider(QgsNativeAlgorithms()
    )
```

```
14
15 from qgis.processing import *
16 import processing
17 from processing.core.Processing import Processing
18
19 from ViewshedAnalysis.visibility_provider import VisibilityProvider
20
21 provider = VisibilityProvider()
22
23 QgsApplication.processingRegistry().addProvider(provider)
24
25 provider.loadAlgorithms()
26
27 Processing.initialize()
28
29 #for alg in QgsApplication.processingRegistry().algorithms():
30         #print(alg.id(), "->", alg.displayName())
```

Listing B.15: Database Connection

```
1
2 # DATABASE
3 conn = sqlite3.connect('raffaele.db')
4 cursor = conn.cursor()
5
6 cursor.execute('SELECT * FROM DATA')
7 data_from_db = cursor.fetchall()
8
9 conn.close()
```

Listing B.16: QGIS Call

```
1
2 #ALGORITHM
3 for row in data_from_db:
4
5         outputs = {}
6
7         #STREETS
8         outputs['CreaPuntiCentroidiDelPixelLungoLaLinea'] =
    processing.run("qgis:generatepointspixelcentroidsalongline", {'
    INPUT_RASTER':'C:/Users/raffa/Desktop/Qgis/w50540_s10.tif','
    INPUT_VECTOR':'C:/Users/raffa/Documents/STRADE.shp|layername=
    STRADE','OUTPUT':QgsProcessing.TEMPORARY_OUTPUT})
```

```
9         outputs['Str'] = processing.run("visibility:createviewpoints"
    , {'OBSERVER_POINTS': outputs['
    CreaPuntiCentroidiDelPixelLungoLaLinea']['OUTPUT'], 'DEM':'C:/Users
    /raffa/Desktop/Qgis/w50540_s10.tif','OBSERVER_ID':'','RADIUS':row
    [2],'RADIUS_FIELD':'','OBS_HEIGHT':1.6,'OBS_HEIGHT_FIELD':'','
    TARGET_HEIGHT':0,'TARGET_HEIGHT_FIELD':'','RADIUS_IN_FIELD':'','
    AZIM_1_FIELD':'','AZIM_2_FIELD':'','ANGLE_UP_FIELD':'','
    ANGLE_DOWN_FIELD':'','OUTPUT':QgsProcessing.TEMPORARY_OUTPUT})
10
11        #POINT
12        outputs['PuntoDiOsservazione'] = processing.run("native:
    pointtolayer", {'INPUT':row[0],'OUTPUT':'C:/Users/raffa/Desktop/
    PUNTO.gpkg'})
13        inp = outputs['PuntoDiOsservazione']['OUTPUT']
14
15        if row[1] != 0:
16                outputs['Buffer'] = processing.run("native:buffer", {
    'INPUT':inp,'DISTANCE':row[1],'SEGMENTS':5,'END_CAP_STYLE':0,'
    JOIN_STYLE':0,'MITER_LIMIT':2,'DISSOLVE':False,'SEPARATE_DISJOINT'
    :False,'OUTPUT':QgsProcessing.TEMPORARY_OUTPUT})
17                outputs['GeneraPuntiCentroidiDeiPixelDentroPoligoni']
     = processing.run("native:
    generatepointspixelcentroidsinsidepolygons", {'INPUT_RASTER':'C:/
    Users/raffa/Desktop/Qgis/w50540_s10.tif','INPUT_VECTOR':outputs['
    Buffer']['OUTPUT'],'OUTPUT':QgsProcessing.TEMPORARY_OUTPUT})
18                inp = outputs['
    GeneraPuntiCentroidiDeiPixelDentroPoligoni']['OUTPUT']
19
20        outputs['Punto'] = processing.run("visibility:
    createviewpoints", {'OBSERVER_POINTS':inp,'DEM':'C:/Users/raffa/
    Desktop/Qgis/w50540_s10.tif','OBSERVER_ID':'','RADIUS':row[2],'
    RADIUS_FIELD':'','OBS_HEIGHT':1.6,'OBS_HEIGHT_FIELD':'','
    TARGET_HEIGHT':0,'TARGET_HEIGHT_FIELD':'','RADIUS_IN_FIELD':'','
    AZIM_1_FIELD':'','AZIM_2_FIELD':'','ANGLE_UP_FIELD':'','
    ANGLE_DOWN_FIELD':'','OUTPUT':QgsProcessing.TEMPORARY_OUTPUT})
21
22        #INTERVISIBILITY
23        processing.run("visibility:intervisibility", {'
    OBSERVER_POINTS':outputs['Punto']['OUTPUT'],'TARGET_POINTS':
    outputs['Str']['OUTPUT'],'DEM':'C:/Users/raffa/Desktop/Qgis/
    w50540_s10.tif','WRITE_NEGATIVE':False,'USE_CURVATURE':True,'
    REFRACTION':0.13,'OUTPUT':'C:/Users/raffa/Desktop/INTER.gpkg'})
```

Listing B.17: QGIS Closure

```
1
2 # Exit QGIS application
3 qgs.exitQgis()
```

Bibliography

[1] *Attività di approntamento*. Oct. 2023. URL: https://www.esercito.difesa.it/Rapporto-Esercito/Mantenere-la-prontezza-operativa-dello-strumento/Addestramento/Pagine/Attivita-di-Approntamento.aspx (cit. on p. 4).

[2] *Onfield Planning*. Nov. 2023. URL: https://www.esercito.difesa.it/comunicazione/pagine/campo-primaverile-per-il-corso-lealta-160510.aspx (cit. on p. 7).

[3] *Map*. Nov. 2023. URL: https://dictionary.cambridge.org/dictionary/english/map (cit. on p. 12).

[4] *Raster Vector*. Nov. 2023. URL: https://www.vebuso.com/2019/01/vector-raster-tale-two-spatial-data-types/ (cit. on p. 15).

[5] *DTM*. Nov. 2023. URL: https://innoter.com/en/services/photogrammetry/digital-terrain-models_dtm/ (cit. on p. 17).

[6] *DSM*. Nov. 2023. URL: https://innoter.com/en/products/spatial-data/dsm-generation/ (cit. on p. 17).

[7] *CTR*. Nov. 2023. URL: https://geoportale.regione.emilia-romagna.it/catalogo/dati-cartografici/cartografia-di-base/cartografia-tecnica/layer (cit. on p. 23).

[8] *QGIS*. Sept. 2023. URL: https://en.wikipedia.org/wiki/QGIS (cit. on pp. 27, 35).

[9] *Discover QGIS*. Oct. 2023. URL: https://www.qgis.org/en/site/about/index.html (cit. on p. 27).

[10] *OOGC*. Nov. 2023. URL: https://commons.esipfed.org/node/362 (cit. on p. 28).

[11] *Open Licence*. Nov. 2023. URL: https://en.wikipedia.org/wiki/GNU_General_Public_License (cit. on p. 28).

[12] Murat Karagamp;ouml;zgil. *Software licenses on github: Which one should you choose?* Nov. 2023. URL: https://muratkaragozgil.medium.com/software-licenses-on-github-which-one-should-you-choose-3d4cfbb6c2f9 (cit. on p. 29).

[13] *QGIS Visibility Analysis*. Nov. 2023. URL: https://www.zoran-cuckovic.from.hr/QGIS-visibility-analysis/help_qgis3.html (cit. on p. 36).

[14] *Viewpoint Logic*. Nov. 2023. URL: https://www.researchgate.net/figure/Ray-tracing-for-visibility-analysis_fig2_277676814 (cit. on p. 37).

[15] *QGIS Grapic Modeler*. Nov. 2023. URL: https://docs.qgis.org/3.4/it/docs/user_manual/processing/modeler.html (cit. on p. 37).

[16] *Flutter*. Sept. 2023. URL: https://flutter.dev/ (cit. on p. 60).

[17] *SQLite*. Nov. 2023. URL: https://www.sqlite.org/about.html (cit. on p. 60).