

POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione



Master of Science in Computer Science and Engineering

Data Processing Pipeline for Augmented Reality Mobile Applications

Supervisor: Prof. Piero Fraternali

Master Graduation Thesis by:

Luca Riva ID. 849976

Jacopo Maria Strada ID. 852252

Academic Year 2016/2017

Abstract

Today it is almost impossible to imagine a world which does not rely on technology on a daily basis. Computers and, lately, smartphones are used by society to communicate, trade and for business and politics. Communication, in particular, allowed people from all the world to exchange knowledge in a simple way, giving life to projects like Wikipedia. For the geographic scope, OpenStreetMap is a project that tries to collect all the knowledge of its users, creating a collaborative dataset comprehensive of a great amount of GIS (Geographic Information System) data. Moreover, agencies with the likes of NASA worked hard to publicly provide topographic data of our planet. This thesis describe a process that aims at processing, improving and creating geographic data and use them for an augmented reality application, exploiting, but also contributing to, such open data sets as Open Street Map.

The proposed data processing and visualization pipeline integrates geo-referenced data collected by multiple sources, extracts relevant information from the various sources, verifies and improves their quality and quantity. These resulting data sets constitute the basis for the augmented reality application. As a prominent original contribution, the pipeline comprises a module that implements a novel algorithm for the discovery of new GIS data, which automatically finds peaks from low-level Digital Elevation Models (DEM) data and then submits the discovered data to a crowdsourcing tool allowing the users to refine the algorithm results.

A second major component of the developed GIS data processing pipeline is the extraction, geo-registration, and visualization of POIs (Points of interest) in an augmented reality mobile application. This part of the work relied on an AR application called PeakLens developed by Politecnico di Milano, which allows users to identify peaks in real time through their smart

phone camera. New elements, such as huts, were extracted from open Street Map and added to the rendered elements of PeakLens, studying a way to represent different elements in the user interface. This required the extension of the app data structure to support non punctual elements in an AR visualization.

Dealing with large and constantly increasing amount of data can be challenging, thus the work performed in the thesis to acquire and analyze data has also paid attention to performance, hardware limitations, and scalability.

Sommario

Al giorno d'oggi è quasi impossibile immaginare un mondo dove la tecnologia non sia presente in svariati ambiti della nostra quotidianità. Computer e, negli ultimi anni, Smartphone sono utilizzati dalla società per comunicare e commerciare, per attività economiche e politiche. Il contributo di internet alla comunicazione, in particolare, ha permesso a persone da tutto il mondo di condividere queste conoscenze in un modo semplice ed immediato, dando origine a progetti come Wikipedia. Dal punto di vista della geografia, OpenStreetMap è un progetto che mira a raccogliere tutti gli elementi geografici che i suoi utenti conoscono, in questo modo la sua comunità è arrivata a creare un enorme *database* di dati GIS (Geographic Information Systems). In aggiunta a questo, enti del calibro della NASA si sono impegnati a raccogliere e condividere pubblicamente dati topografici del nostro pianeta.

La procedura di elaborazione e visualizzazione che viene proposta in questo lavoro integra dati geo-referenziati provenienti da diverse risorse, estraendo informazioni rilevanti da ognuna di esse, verificandone la qualità e arricchendole. I dati che vengono ottenuti al termine di questa procedura sono alla base dell'applicazione di realtà aumentata. Come originale contributo, questa procedura comprende un modulo dedicato all'implementazione di un nuovo algoritmo per la scoperta di nuovi dati GIS, in particolare con lo scopo di trovare nuovi picchi a partire da DEM (Digital Elevation Models) di basso livello e successivamente riportare questi risultati in uno strumento di *crowd-sourcing* che consente agli utenti di controllare i risultati dell'algoritmo.

La procedura di elaborazione dei dati GIS implementata ha un altro grande componente che si occupa di estrazione, geo-registrazione e visualizzazione di POI (Points of Interest) in un'applicazione di realtà aumentata. Questa parte del lavoro si fonda sul lavoro svolto dal Politecnico di Milano nella realizzazione di un'applicazione di realtà aumentata chiamata PeakLens, che

consente all'utente di identificare picchi montuosi in tempo reale attraverso ciò che viene inquadrato dalla fotocamera dello smartphone. Nuovi elementi, come i rifugi, sono stati importati da OpenStreetMap e aggiunti agli elementi rappresentati da PeakLens, dovendo quindi studiare un modo di integrare diversi elementi nell'interfaccia dell'utente. Questo ha richiesto anche l'estensione della struttura dati dell'applicazione per arrivare al supporto di elementi non puntuali nella visualizzazione in realtà aumentata.

Lavorare con una grande mole di dati in continuo sviluppo può essere molto complicato, di conseguenza, in questa tesi, nell'implementare di procedure d'importazione e analisi dei dati è sempre stata rivolta una grande cura alle performance, alle limitazioni hardware e alla scalabilità

Acknowledgements

First of all, we would like to thank our supervisor, Piero Fraternali, for having given us the opportunity to take part into this exciting and interesting project. His support during every stage encouraged us to work with a positive attitude. Furthermore, we are grateful to him and his research group for their research work in the field of augmented reality relative to the mountains environment, which both of us love.

I would also like to thank my parents, Marco and Maria Angela, and all my grandparents, Carlo, Germano, Laura and Palmira, for being extraordinarily supportive and giving me warm encouragement . A special thank is reserved for my love, Roberta, that was always present throughout my years of study and during the preparation of this thesis.

Thank you.

Luca

Finally, I must express my very profound gratitude to my parents, Massimo and Cristina, to my brother Tommaso and to my grandparents Pinuccio and Lina for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of implementing and writing this thesis work. I also would like to thank all my friends, I do not know how you put up with me for all these years, but I would not have reached this goal if it weren't for you.

Thank you.

Jacopo

Contents

1	Introduction	1
1.1	Background	2
1.1.1	PeakLens and SnowWatch	2
1.1.2	Open Street Map and Overpass API	3
1.1.3	Wikidata	5
1.1.4	Augmented Reality	5
1.1.5	Geomorphometry	6
1.2	Achieved Results	8
1.3	Document Structure	8
2	Related Work and State of the Art	9
2.1	Augmented Reality	9
2.1.1	Augmented Reality Applications	9
2.2	Morphometric Feature Extraction from DEM	10
2.2.1	Multi-scale fuzzy method	12
2.2.2	Hydrological stream extraction workflows	13
2.2.3	Multi-method procedures	13
3	Problem Statement and Proposed Approach	15
3.1	Data management	20
3.1.1	Data types	20
3.1.2	Data sources selection	22
3.1.3	POI importing process	25
3.2	Augmented reality rendering of monodimensional elements	31
3.2.1	Identification of position and orientation	31
3.2.2	Virtual Panorama and Visibility	32
3.2.3	Pagination	33
3.2.4	Alignment of object positions with the camera view	37
3.2.5	Object Visualization in the Camera View	40
3.3	Automatic peaks extraction from DEM data	42

3.3.1	Detection of Ridges	45
3.3.2	Detection of Peaks	48
4	Implementation Details	51
4.1	Data retrieval	53
4.1.1	OpenStreetMap and Overpass API	53
4.1.2	WikiData	57
4.1.3	Stream approach	58
4.1.4	PeakLens Data Structure	60
4.2	Augmented reality rendering of monodimensional elements	61
4.2.1	PeakLens Render API	61
4.2.2	Virtual Panorama Downloader	64
4.2.3	Visualization	65
4.3	Peaks extraction from DEM	67
4.3.1	HGT files	67
4.3.2	Fuzzy Values Computation	68
4.3.3	Grouping	71
4.3.4	Data Storage	72
5	Evaluation	73
5.1	Alpine huts integration in Peaklens	74
5.1.1	Quantitative Evaluation	74
5.1.2	Qualitative Evaluation	77
5.2	Peaks discovery	79
5.2.1	Improvements achieved over the baseline method	79
5.2.2	Performance Evaluation	81
6	Conclusions and Future Work	87
6.1	Conclusions	87
6.2	Future work	88
6.2.1	Managing non punctual elements	89
6.2.2	Applying other algorithms to extract peaks from DEM	90
6.2.3	Discover new peak names through crowdsourcing	90
6.2.4	Improving data directly on OSM	91
	Bibliography	93

List of Figures

1.1	A PeakLens screenshot	3
1.2	OSM data in the Grigna Group area	4
1.3	Feature extraction from DEM	7
2.1	Morphometric Classes	11
2.2	Fuzziness of morphometric classes	12
3.1	Heterogeneous geo-referenced data management overview	15
3.2	PeakLens data management overview	17
3.3	Augmented Reality Visualization Example of different types of elements	21
3.4	Monte Bianco Names	24
3.5	POI importing process	26
3.6	Elevation errors	28
3.7	Elevation Check Process	29
3.8	Orientation Sensors Calibration	31
3.9	Virtual Panorama	32
3.10	Sensors data. Figure taken from [26]	38
3.11	Comparison between sensor and global matching	39
3.12	Main interface of PeakLens showing labels before (a) and after (b) the restyling	40
3.13	Flowchart diagram of peaks extraction defined in [23]	42
3.14	Overall process designed to find peaks	44
3.15	Flowchart diagram of ridges extraction	45
3.16	Windows of cells of increasing size for fuzzy value computation	46
3.17	Representation on a map of fuzzy values denoting ridges	47
3.18	Flowchart diagram of groups creation	48
3.19	Representation on a map of fuzzy values denoting peaks	49
3.20	Peaks positioning within groups of cells	50
4.1	Main Components of the system	51

4.2	PeakLens Client-Server Infrastructure	61
4.3	PeakLens Render API Class Diagram	62
4.4	PeakLens App Pois Class Diagram	64
4.5	PeakLens App Drawers Class Diagram	65
4.6	HGT Structures Example	68
5.1	Examples of ground truth images, extracted by an evaluator from the sequences recoded in the field	75
5.2	Assessment Questionnaire Results	78
5.3	Improvements in peaks recognition on ridges	79
5.4	Improvements in peaks recognition in non-mountain areas . .	80
5.5	Precision-Recall Curve (configurations and values are shown in Table 5.2)	82
5.6	Area of extraction - Around Everest, Himalaya	84
5.7	Expert Crowdsourcing Interface	85
6.1	Possible approach to non punctual elements label positioning	89
6.2	Social Media Crowdsourcing Mockup	91

List of Tables

3.1	Types of elements that can be displayed in augmented reality	20
3.2	Huts from the Italian Alps regions	23
3.3	Elevation difference between DEM and OSM	30
3.4	Result of the pagination when peaks and huts are treated as the same type of element	34
3.5	Result of the pagination done with the multiple steps algorithm with bindings creation	37
5.1	Average Degree Error	76
5.2	Values used for Figure 5.5 (See section 3.3 for their description)	82
5.3	Extraction statistics using strict parameters (Configuration A of Table 5.2)	83
5.4	Extraction statistics using loose parameters (Configuration J of Table 5.2)	83
5.5	Extraction statistics using strict parameters for Everest area, Himalaya	84
5.6	Strict extraction parameters for Everest area, Himalaya (See section 3.3 for their description)	84
5.7	Facebook Groups Statistics	86

List of Algorithms

- 3.1 Peaks Pagination 33
- 3.2 Bindings Creation 35
- 3.3 Huts Pagination 36

List of Listings

4.1	Example of hut element from OSM in XML format	54
4.2	Query to retrieve all the alpine hut elements from the Overpass instance	55
4.3	Overpass response example	56
4.4	Example response from WikiData	57
4.5	GeoJSON Example	59
4.6	JSON Stream usage example	60
4.7	PeakLens data format	60
4.8	PeakLens Render API Response	62
4.9	Poi Drawer	66
4.10	Function used to compute fuzzy values for ridges and peaks .	69
4.11	Functions used to create groups of cells	71

Chapter 1

Introduction

This thesis work describes an approach to improve the quality of heterogeneous geo-referenced data and their display in location-dependent mobile applications. In particular a possible contribution to these fields is discussed proposing an original mix of automated algorithms allowing to import a variety of geographical data and keep them updated, render them in an augmented reality mobile application, discover missing data and take advantage of crowdsourcing in order to validate and complete these data.

SnowWatch deeply inspired this work and also provided an important technological and innovative contribution in the field of environmental monitoring. The project was started by the Politecnico di Milano few years ago and is still under development. In this chapter an overview of the project is presented, the scientific and technological innovations introduced and the link between SnowWatch and this thesis work analyzed.

Many are the areas touched during the development of this work, among them data management (acquisition from multiple sources, validation, completion), augmented reality rendering of geographic features and geomorphometry for the analysis of peaks areas.

In this chapter the various sectors involved will be introduced and discussed in a broader sense to give an idea of the context in which the work is centered. More details will be given in the next chapters.

1.1 Background

1.1.1 PeakLens and SnowWatch

SnowWatch is an environmental monitoring project that has been funded by the European Union and carried out by the Web, Data and Society research group of the Politecnico di Milano.[12] This project aims at monitoring the amount of water available on the mountains detecting the snow reserves among them. In order to do this, SnowWatch takes advantage of user generated photos taken from social networks combined with satellite photos and photos taken from webcams placed in several mountain areas. This is possible thanks to algorithms that are able to analyze the content of these photos and extract relevant information. This project also presents a way to extract the skyline from a mountain panorama distinguishing the skyline from obstructing objects or clouds. This algorithm allowed to extract a skyline representation of a geolocated photo and compare it to a tridimensional representation of the visible panorama from the point in which the photo has been taken and to obtain parameters like the camera orientation and field of view with a good approximation. The tridimensional model of the Earth surface is built by using the DEM (Digital Elevation Model), that is a collection of latitude, longitude and altitude values detected by satellites in a grid of points uniformly spaced on the Earth's surface. Once the estimation of orientation and field of view is ready, the system determines visible peaks positions and adds tags with the names of the peaks in the detected positions.

Later it has been decided to give users an active role in collecting photos and therefore to develop a mobile application that, by using the augmented reality as a means of displaying information, provides the use of the peak identification service explained above in real time. From this idea the application for Android devices, named PeakLens, was born.[22] PeakLens allows to identify peaks in real time using frames from the smartphone camera, smartphone's GPS position, camera orientation, gyroscope, magnetometer and accelerometer. With these data PeakLens is able to provide a great precision for the labels it places on the panorama, giving a pleasant and immersive experience to the end users. This also enables PeakLens capacity of producing content suitable to environmental monitoring systems. Photographs generated through PeakLens have everything useful for this task: they comprehend metadata on location, orientation and field of view ob-

tained through sensors. In turn sensors are subsequently improved through the skyline extraction and alignment algorithms, which overcome sensor errors and inaccuracies [26].



Figure 1.1: Augmented reality labels from PeakLens (Figure taken from [10])

An application like PeakLens deeply relies on comprehensive and updated data, the presented work proposes a system to integrate a large number of elements in PeakLens with a simple and effective way to keep them updated. At the same time the aim is to systematically find new elements that are not provided by the traditional sources and add them to the application.

1.1.2 Open Street Map and Overpass API

OpenStreetMap (OSM) is a free, editable map of the whole world that is being built by volunteers largely from scratch and released with an open-content license. The OpenStreetMap License allows free (or almost free) access to map images and of all underlying map data. The project aims to promote new and interesting uses of this data. Map's modification happens in a wiki-like fashion: users can count on tools and API in order to freely edit the data.

The OpenStreetMap Foundation is an organization that performs fundraising. One major expense is acquiring and maintaining the servers that host the OpenStreetMap project. While the foundation supports the project,

it does not control the project or "own" the OSM data. The foundation is dedicated to encouraging the growth, development and distribution of free geospatial data and to providing geospatial data for anyone to use and share.[6]

The Overpass API (formerly known as OSM Server Side Scripting, or OSM3S before 2011) is a read-only API that serves up custom selected parts of the OSM map data. It acts as a database over the web: the client sends a query to the API and gets back the data set that corresponds to the query.

Unlike the main API, which is optimized for editing, Overpass API is optimized for data consumers that need a few elements within a glimpse or up to roughly 10 million elements in some minutes, both selected by search criteria like e.g. location, type of objects, tag properties, proximity, or combinations of them. It acts as a database back-end for various services.

It is also possible to install and run overpass on a proprietary server as it is licensed under the Affero GPL v3. [8]

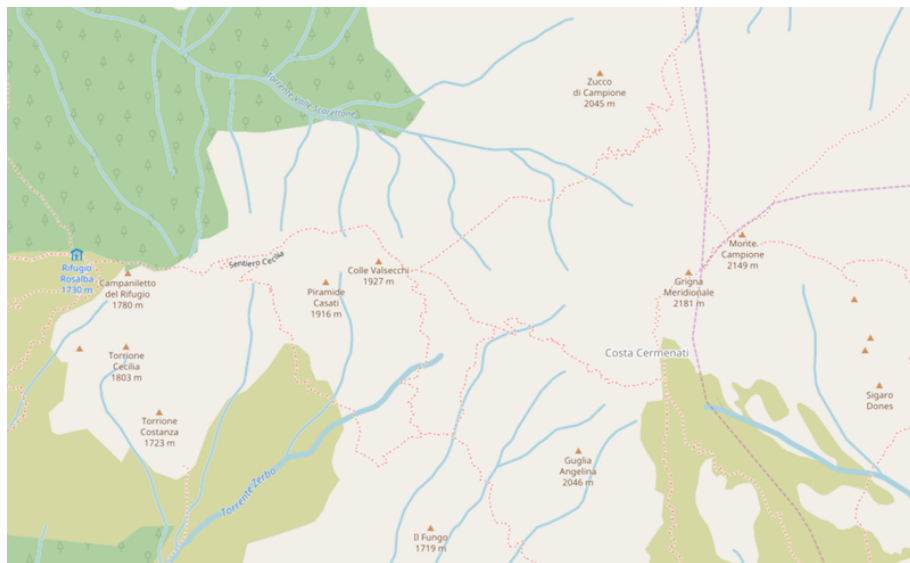


Figure 1.2: Data from OSM represented on the map: all this data are openly available via API and provide information like coordinates and elevation, essential for PeakLens

Thanks to its richness of geographic data and its open content license, OSM has been largely used in this work in order to integrate different elements like peaks and huts and will also be a great candidate for future improvements.

1.1.3 Wikidata

Wikidata is a free and open knowledge base that can be read and edited by both humans and machines. Wikidata acts as central storage for the structured data of its Wikimedia sister projects including Wikipedia, Wikivoyage, Wikisource, and others. [18]

Wikidata also provides support to many other sites and services beyond just Wikimedia projects! The content of Wikidata is available under a free license, exported using standard formats, and can be interlinked to other open data sets on the linked data web.

The data in Wikidata is published under the Creative Commons Public Domain Dedication 1.0, allowing the reuse of the data in many different scenarios. It is possible to copy, modify, distribute and perform the data, even for commercial purposes, without asking for permission.

Editing, consuming, browsing, and reusing the data is fully multilingual. Data entered in any language is immediately available in all other languages. Editing in any language is possible and encouraged.

Imposing a high degree of structured organization allows for easy reuse of data by Wikimedia projects and third parties, and enables computers to process and "understand" it: the Wikidata repository consists mainly of items, each one having a label, a description and any number of aliases. Items are uniquely identified by a Q followed by a number, such as Mount Everest (Q513).

Statements describe detailed characteristics of an Item and consist of a property and a value. Properties in Wikidata have a P followed by a number, such as with topographic prominence (P2660).[17]

Wikidata offers many kinds of data, in this work its information about geographical elements has been used to improve the ones imported from OSM.

1.1.4 Augmented Reality

Outdoor augmented reality applications exploit the position and orientation sensors of mobile devices in order to estimate the location of the user and

her field of view so as to overlay such view with information pertinent to the user's inferred interest. These solutions are finding a promising application in the tourism sector, where they replace traditional map-based interfaces with a more sophisticated user experience whereby the user automatically receives information based on what he is looking at, without the need of manual search. Examples of such AR apps include, e.g, Metro AR and Lonely Planet's Compass Guides. The main challenge of such applications is providing an accurate estimation of the user's current interest, adapted in real-time to the changing view. Most commercial applications simplify the problem by estimating the user's interest based only on the information provided by the device position and orientation sensors, irrespective of the content actually in view. Examples are sky maps, which show the names of constellations, planets and stars based on the GPS position and compass signal. An obvious limit of these approaches is that they may provide information that does not match well what the user is seeing, due to errors in the position and orientation estimation or to the presence of objects partially occluding the view. These limitations, besides jeopardizing the user's experience, prevent the possibility for the AR application to create augmented content. If the overlay of the meta-data onto the view is imprecise, it is not possible for the user to save a copy of the augmented view, e.g., in the form of an image with captions associated to the objects. Such augmented content could be useful for several purposes: archiving the augmented outdoor experience, indexing visual content for supporting search and retrieval of the annotated visual objects, and even for the extraction of semantic information from the augmented content [22].

In the following sections it is explained how the algorithms implemented in PeakLens have been used in order to place new elements and enrich the user experience.

1.1.5 Geomorphometry

Geomorphometry is the science of quantitative land-surface analysis. It evolved directly from geomorphology and quantitative terrain analysis, two disciplines that originated in 19th century geometry, physical geography, and the measurement of mountains. Modern geomorphometry addresses the refinement and processing of elevation data, description and visualization of topography, and a wide variety of numerical analyses. The operational goal of geomorphometry is extraction of measures and spatial features from

digital topography. The usual input to geomorphometric analysis is the digital elevation model (DEM) [28].

The DEM data structure represents locations attributed with elevation values organized spatially as a raster, a regular grid of points or cells. Such representations of terrain allow topographic analysis across the entire spatial extent of the model [29]. This structure is especially advantageous to geomorphometry because most of its technical properties are controlled automatically by a single measure: grid resolution (i.e. 3-arc-second or 1-arc-second). In addition to grid resolution, the coordinates of at least one grid intersection (usually marking the lower left-hand corner of the entire DEM array) and the number of rows and columns are needed, whereupon it should be possible to define the entire map [28].

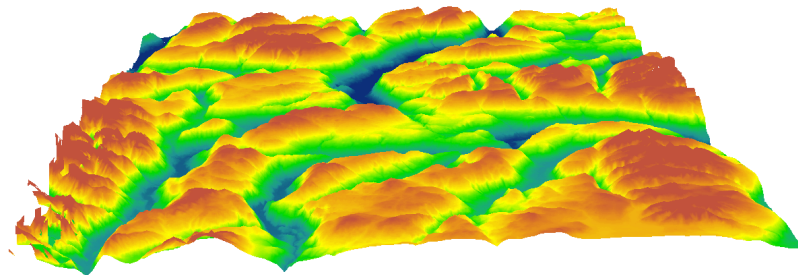


Figure 1.3: A deep analysis of a Digital Elevation Model allows the extraction of different features of the terrain (e.g. peaks, ridges...)

Starting from a digital elevation model, in this thesis geomorphometry is approached focusing on the extraction of spatial features. The inspiration is taken from the fact that the landscape in which people live is made up of many features, which are named and have importance for cultural reasons but in the open cartography that is publicly available often a lot of features are missing. Prominent among these are the naming of upland features such as mountains [23] and it is relevant to study how to contribute to the enrichment of peaks data.

1.2 Achieved Results

With a new implementation of the process to retrieve data from multiple data sources (OSM, WikiData, DEM) it was possible to improve the amount and the quality of data, collecting the elevation for elements which had no elevation specified in OSM and filtering out the ones that had wrong elevations. At the same time it was possible to add alternative names to the elements to improve usability for more users speaking different languages.

Adding another kind of elements, i.e. peaks, PeakLens provides a richer experience to the users: a few testers were asked to try the new version of the application and were generally satisfied with the results.

Peaks extraction brought interesting results at a first glance, a crowdsourcing tool is ready to test the algorithm showing its results to mountain experts that are enabled to confirm or discard the extracted peaks.

1.3 Document Structure

In [chapter 2](#) the state of the art of the most important areas involved in this work is presented together with some relevant applications.

In [chapter 3](#) the designed pipeline is present starting from a general case and specializing it to the real implemented case explaining all the modules created or modified in the context of this work.

In [chapter 4](#) the architecture of the system is described and the details about all the modules are explained.

In [chapter 5](#) is presented a qualitative and quantitative evaluation of the main modules of this thesis.

Finally in [chapter 6](#) the conclusions are exposed and some considerations about the possible future improvements of this works are presented.

Chapter 2

Related Work and State of the Art

2.1 Augmented Reality

Augmented reality is a variation of virtual environments, or virtual reality as it is more commonly called. Virtual reality technologies completely isolate the user from the real world. The user, while immersed in a synthetic environment, is not able to see the real world around him. In contrast, augmented reality allows the user to mix the real world and the virtual objects. This is achieved by superimposing virtual objects to real-time images of the world. Therefore, augmented reality supplements reality, rather than completely replacing it. Ideally, it would appear to the user that the virtual and real objects coexisted in the same space.[19]

2.1.1 Augmented Reality Applications

Augmented reality is a well established research topic within the Human Computer Interaction field, which has attracted new attention thanks to the announcement by major hardware vendors of low-cost, mass-market augmented reality devices. In particular, the recent trend of using smartphones and tablets as augmented reality platforms benefits from the improved standardization (most augmented reality software can now be used without ad hoc hardware), increased computational power and sensor precision. The

survey in [21] overviews the history of research and development in augmented reality, introduces the definitions at the base of the discipline, and positions it within the broader landscape of other technologies. The authors also propose design guidelines and examples of successful augmented reality applications and give an outlook on future research directions. An important branch of the discipline is the outdoor augmented reality. Several works address the problem, usually to identify and track points of interest in urban scenarios. Although standard solutions for mobile AR already exist (e.g. Wikitude2), they rely only on compass sensors or the a priori known appearance of the objects [22]. Examples of other Apps in this field are ViewRanger and PeakFinder, two mobile applications for Android and iOS. PeakFinder uses elevation models integrated in the App in order to draw a 360° panorama for a given point of view and assigns labels with names and basic info to the peaks. Recently, PeakFinder also started giving the possibility to overlap its drawn panorama on the images retrieved from the camera [9]. ViewRanger, among many other features like route guides, maps and navigation, also offers a feature called Skyline which adds labels in real time to the images from the smart phone camera using GPS and compass to align them on the relative elements (e.g peaks, lakes, towns) [15][14]. The main difference between these two applications and PeakLens is that ViewRanger and PeakFinder only use phone sensors in order to apply labels on the elements, while the latter use sensors and an AI algorithm in order to obtain a more precise alignment.

2.2 Morphometric Feature Extraction from DEM

The attempt to compute the landscape has given rise to an area of research known as geomorphometry; the measurement of the shape of the earth's surface. Two approaches are descriptive of geomorphometry: the generation of descriptive statistics of the shape of the surface and the assignment of a location in a landscape to an exhaustive set of features based on the local form of the land surface. Among the simpler, geometric (and therefore computable) set of forms is the assignment of a location to one of six morphometric classes: pit, peak, pass, channel, ridge and plane. These six classes are illustrated in Figure 2.1, and have very obvious correspondence with the expected form of features which people recognize in the landscape, namely, peak with mountain, channel with valley, and ridge and pass are often named as such.

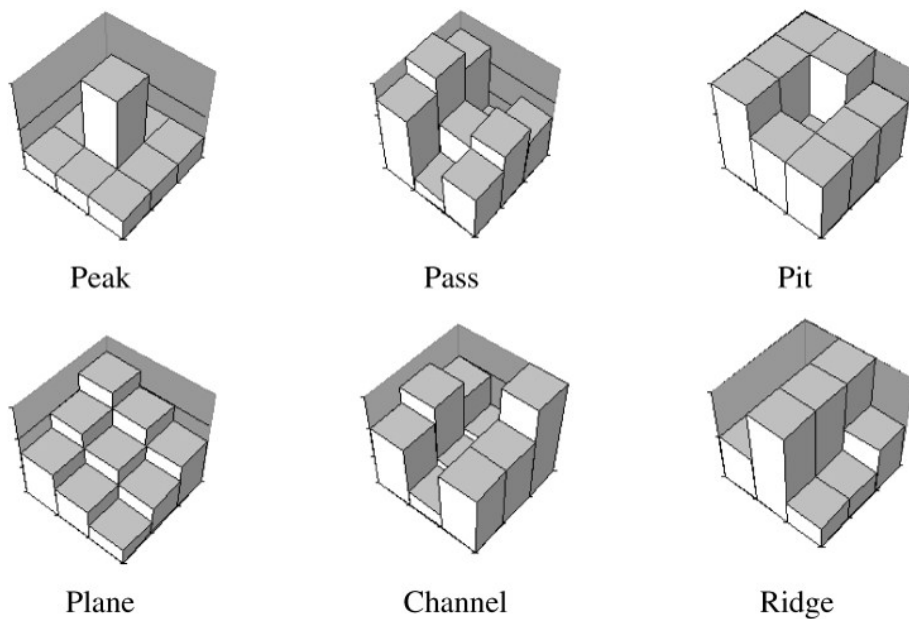


Figure 2.1: The six morphometric classes as represented by a gridded elevation model (Figure taken from [23])

There is in fact a precise definition of the different morphometric classes, and so any location can be allocated to a specific class, but the class to which a location is assigned by this precise process varies due to the scale of measurement giving rise to ambiguity as to the correct classification and so vagueness.

The idea of vagueness in geography has been told to be appropriate for the analysis by fuzzy sets from many researchers. Classic set theory, also known as Boolean set theory, assign an inter value of 1 as membership of a set if an object belongs to that set. If it does not belong, a membership of 0 is assigned. Instead, in fuzzy set theory, a core concept is defined and objects which exactly match that core concept are assigned a class membership value of 1. The membership is assigned a reducing real number for objects as they are increasingly dissimilar from that core concept until they have no similarity to the core concept, when the membership is assigned a value 0.

In the case of morphometry, however, one reason the morphometric class is vague is the geographical scale of measurement. By scale, is meant a combination of both spatial extent and spatial detail or resolution, and in the explained research, is used the variation in the extent over which the feature is defined as the basis of the fuzzy membership.

Due to this fuzziness, what may be a channel at one scale may be another morphometric class at another scale, say a ridge (see Figure 2.2). So is it a channel or a ridge? Clearly it is to some degree both. Indeed, as demonstrated by Wood in [30], many locations may be classed as all of the above-mentioned six morphometric classes, depending solely on the scale of measurement [23].

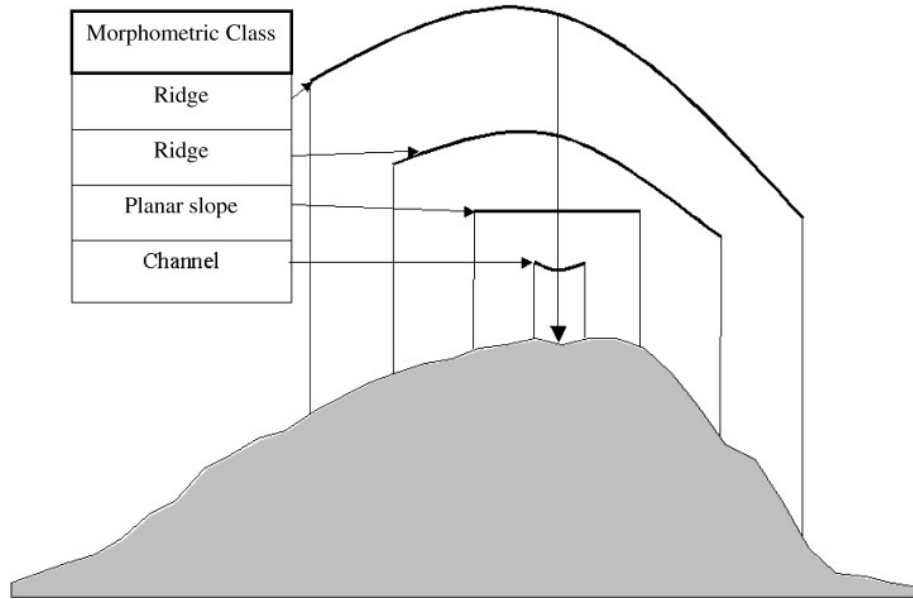


Figure 2.2: The morphometric class at the point indicated by the vertical arrow varies as shown with the scale over which it is measured. (Figure taken from [23])

2.2.1 Multi-scale fuzzy method

Starting from the idea explained before, in order to overcome the problem that a location may be classified as all of the six morphometric classes, a method that considers more than one scale has been developed. Considering that at any particular scale the landscape can be assigned to one morphometric class as a Boolean or crisp set, the vagueness of the assignment of any location to a morphometric class is that landform class is not necessarily stable under repeated observation at different scales. If this is the case, then it follows that the fuzzy membership of a morphometric class at location can be given by a weighted average of the Boolean memberships of that class over the scales of measurement [23].

In order to apply this method on DEMs, a moving window of increasing size

in the grid in used to compute the fuzzy value of each point of the DEM. The number of time the window dimension is changed represent how many scales are considered for the process.

2.2.2 Hydrological stream extraction workflows

Hydrological stream extraction workflows can be used to find courses and ridges in DEMs. These methods are often included in the most popular GIS software like ArcGIS, GRASS, SAGA. . . . In these programs, the workflow is usually intended to build linear and irregular stream networks using moving window approaches to sequentially identify cells with the lowest elevations in local cell neighborhoods. Thresholding is applied to these approaches to extract cells with high flow accumulation values, which can introduce bias. For courses, the thresholding step results in some high elevation streams with low flow accumulation values being missed or poorly represented as compared to those associated with low elevations and high flow accumulation values. It is also known that areas found by stream extraction methods will not exactly match streams in the real environment due mainly to their complex nature. To extract ridge lines, the same method can be used, but with the inverted DEMs [29].

2.2.3 Multi-method procedures

Different feature extraction methods typically produce different results, even if they are studied to find similar features. This is due to the fact that each method models features in a slightly different way. Leveraging these dissimilarities is possible to analyze complementary perspectives for understanding a general feature class. It was found that combining results from multiple methods provides a more robust representation of each terrain feature class [29].

Chapter 3

Problem Statement and Proposed Approach

This thesis addresses the general problem of improving the quality of heterogeneous geo-referenced data and their display in location-dependent mobile applications. The approach defined to address such problem exploits a tool chain that supports all the steps of heterogeneous geo-referenced data management: acquisition from multiple sources, validation, completion, fusion, and visualization, by an original mix of automated algorithm (e.g. for discovering missing data) and crowdsourcing (e.g., for validating and completing data).

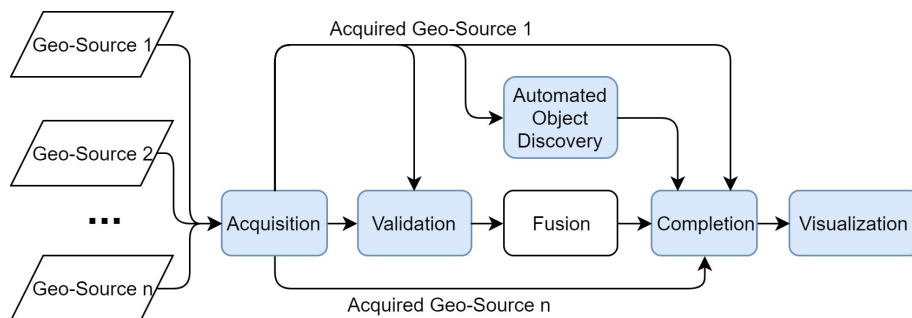


Figure 3.1: Heterogeneous geo-referenced data management overview

Figure 3.1 represents the flowchart of the general process of managing heterogeneous geo-referenced data; its main steps are the following:

Acquisition In this phase data are gathered from various sources through API calls or simple download of datasets.

Validation Data acquired at the previous step are subject to some checks in order to establish their veracity.

Fusion When multiple data sources are used there is the need to merge their data. In this phase objects coming from different sources must be either recognized as the same object and then integrated or recognized as different objects and then added independently.

Completion In this step information coming from side data sources are used to enrich the already present objects.

Visualization Culminating phase in which the elements prepared in the previous steps are shown to the users depending on their location.

Automated Object Discovery This activity is meant to discover new objects analyzing the surface of the earth available as Digital Elevation Model.

In Figure 3.1, a blue background highlights the steps for which this thesis work has provided an original contribution.

Specifically, the thesis applies the aforementioned approach to a case study of geo-referenced data about mountains, including two types of objects: morphological elements (e.g. peaks) and Point of Interests - POIs - (e.g. huts). The implemented tool chain acquires geo-referenced data from multiple data sources (Open Street Map GIS data about peaks and POIs and Digital Elevation Model point cloud about geo-morphology), expands the data (about peaks) with a mixed human and automatic process, and finally visualizes the data about peaks and POIs on the GUI of a mobile outdoor augmented reality App called PeakLens.

Figure 3.2 shows the flowchart representing the process of data management in PeakLens; again, the blue background denotes the steps where this thesis work made a contribution.

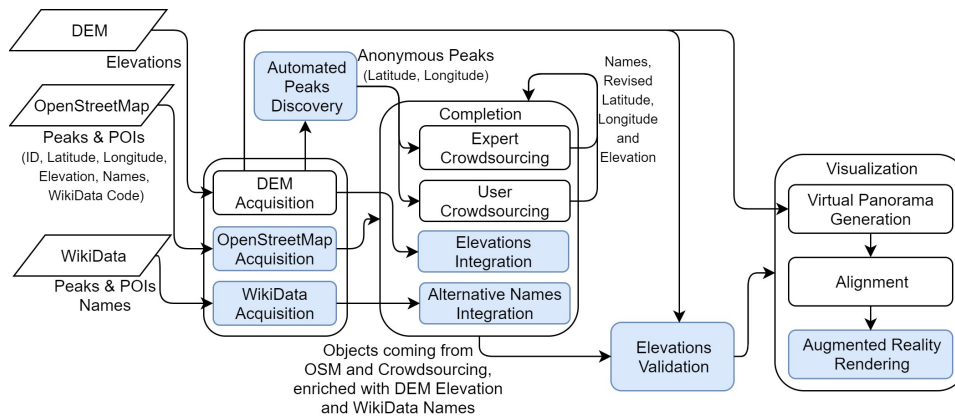


Figure 3.2: PeakLens data management overview

The phases shown in [Figure 3.2](#) are the following:

Acquisition This step is responsible of downloading objects of different type the from OpenStreetMap (OSM) GIS database [6]. Also the Digital Elevation Model (DEM) is acquired from a public source [16].

Automated Peaks Discovery In this phase, different algorithms are used to find new peaks.

Completion In this step the information retrieved using OpenStreetMap is enriched. This is done in different ways:

- Addition of alternative names from a data source complementary to OSM, such as WikiData [18]. This step permits to cope with the case in which the same peak is known under multiple names. Relevant examples are the ones where the original name has a different alphabet in respect of the user’s one (Everest original name is reported in ideograms from OSM, obviously unreadable for most users).
- Insertion of elevations computed from the Digital Elevation Model. This step permits to fill gaps in the peaks and POIs metadata provided by OSM, which may be missing elevation data.
- Insertion of new peaks automatically discovered and then confirmed and named by experts or standard users through crowdsourcing. Contributions may be provided by means of external crowdsourcing services or directly by means of the PeakLens App. This step supports the completion of the peak data set

provided by OSM, which is largely defective in many regions of the world. Comparing Lombardy, which is an area of 24 thousands square meters, and Nepal, which is an area of 147 thousands square meters largely occupied by the Himalaya mountain range, it is possible to notice that, at the time of writing only 181 peaks are stored in OSM for Nepal against 2945 for Lombardy.

Elevation Validation In this phase the elevation data acquired from OSM are compared with the ones computed from the Digital Elevation Model, in the previous step. This procedure allows the discovery of errors in the OSM data.

Visualization This step visualizes the geo-referenced data about peaks and huts. Visualization is performed through a mobile application: PeakLens, which overlays the peaks and huts in the camera view in real time during outdoor usage. The PeakLens visualization module exploits a Virtual Panorama Generation method and Computer Vision techniques to align the image taken from the camera and the Virtual Panorama; the application shows in augmented reality the elements processed in the previous phases over the mountain landscape in the camera view (see Figure 3.12(b) for an example of how PeakLens visualizes peaks and huts).

The work of this thesis is part of a larger research project, aimed at the semi-automatic collection and processing of geo-referenced data for environmental applications [26, 10, 12, 24].

The specific contribution of this thesis in the context of the research project, can be described as follow:

- A new server side module for downloading morphological elements (e.g. peaks) and points of interest (e.g. huts) in an efficient and comprehensive way from OSM. Contribution is described in [section 3.1](#).
- A new server side data completion component, usable to enrich peak objects with alternative names coming from WikiData. This contribution is described in [Section 3.1](#).
- A new server side data validation component, exploiting the Digital Elevation Model, for checking elevation data in order to correct or improve them. This contribution is described in [Section 3.1](#).

- A complete processing pipeline (from acquisition to validation) for novel types of objects to be rendered in an augmented reality fashion within the PeakLens application. As a proof of concept, the pipeline has been applied to mountain huts, chosen as representative for monodimensional points of interest and used in this work. The provided pipeline can be applied to any other class of monodimensional objects, such as churches or castles. This contribution is described in Section [section 3.1](#).
- A completely new server side object discovery module, usable to automatically discover new peaks by analyzing the point cloud of the Digital Elevation Model of the terrain. This module has required the improvement of state of the art algorithms for the morphological analysis of elevation models [23], in order to enhance the accuracy of morphological object extraction. The module scales to the processing of the DEM of the entire Earth. This contribution is described in Section [section 3.3](#).
- A new client side component for visualizing a new class of objects (point of interests of type hut). This contribution is described in Section [section 3.2](#).

3.1 Data management

This section presents the process of handling geo-referenced data, from the selection of a data source to an overview of the saved data structure.

3.1.1 Data types

Before the contribution of this thesis, PeakLens managed only one type of punctual elements: peaks. Nonetheless, geographic elements of interest for an augmented reality application may have different shapes, which cannot be defined by a simple point; relevant examples are elements such as rivers, hiking trails or cities.

Table 3.1 provides a classification of elements that can be displayed in an augmented reality outdoor application. The problem with the inclusion of these elements is that they can be defined by the coordinates of multiple points, which requires extending the existing PeakLens data structure in order to support new types of non-punctual elements.

	Points	Polygons	Polylines
Morphological Elements	Peaks, Volcanos, Passes	Lakes, Glaciers	Streams, Rivers, Valleys
Points Of Interest	Alpine and Wilderness Huts, Castels, Churches	Cities	Footpaths

Table 3.1: Types of elements that can be displayed in augmented reality

Elements such as huts and buildings in general are characterized by the vertex coordinates of the rectangle containing them. For the purpose of augmented reality representation of this kind of elements, it is sufficient to select just one point (e.g., the centre of the rectangle) where the label may be placed.

Conversely, elements occupying a wide area require a data structure able to store multiple coordinates per element and a more complex visualization.

An example of an object requiring the management and visualization of non-punctual data is visible in [Figure 3.3](#).

This work has addressed the extension of PeakLens to manage non punctual objects, by defining a richer data structure to store multi-coordinates elements.

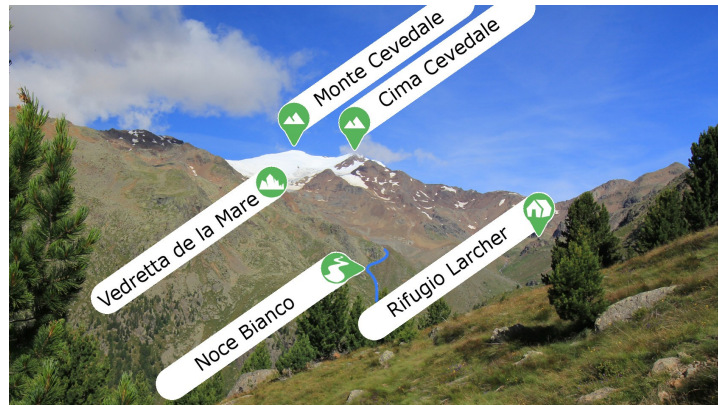


Figure 3.3: Augmented Reality Visualization Example of different types of elements: "Monte Cevedale" and "Cima Cevedale" are peaks and together with "Rifugio Larcher" that is an alpine hut are representative of point elements. "Vedretta de la Mare" is a glacier and is an example of polygonal rendition. "Noce Bianco" is a stream and in this figure represent polyline elements.

3.1.2 Data sources selection

In this work two types of data sources have been used: *GIS Objects data sources* and *Earth Models data sources*. The former are collections of natural and manmade objects supplied with the necessary information to locate them on the earth. The latter consists of all the data necessary to reconstruct the geoid.

Below the details about the selection of these sources can be found.

GIS Objects data sources

The selection of one comprehensive, high quality and publicly available data source was one of the crucial steps of this work. It is possible to find a great and increasing amount of geographic data sources, but it is not easy to find one that covers the whole world: depending on the popularity of the area many local datasets are available, but the global datasets are just a few.

In particular the selection of a data source is driven by the following criteria:

- **Global:** One source that cover all the world
- **Open License:** License that allow the reuse of data
- **API Access:** Easy access to data trough open APIs
- **Presence of Names and Coordinates:** Names and coordinates must be present as fields of the interesting elements

Taking into account huts, one of the elements relevant for the PeakLens context, available data sources were inspected, with a focus on the specific area of Italian Alps, as reported in [Table 3.2](#).

Region	Source	n° of huts	Coordinates	License
Val d'aosta	Montagne Val d'Aosta	68	not always	N/A
	LovevDa	59	yes	N/A
	Rifugi in rete	56	yes	CC
	OSM	75	yes	ODbL
Piemonte	Rifugi Piemonte	45	yes	N/A
	CAI Piemonte	70	no	N/A
	Rifugi in rete	111	yes	CC
	Piemonte outdoor	147	yes	N/A
	OSM	255	yes	ODbL
Lombardia	Rifugi Lombardia	193	yes	N/A
	CAI	75	no	N/A
	Diska	315	yes	N/A
	OSM	313	yes	ODbL
Trentino	Trentino Rifugi	147	no	N/A
	Visit Trentino	140	yes	N/A
	OSM	143	yes	ODbL
Sud Tirolo	Malghe e Rifugi	174	yes	N/A
	OSM	218	yes	ODbL
Veneto	CAI Veneto	108	yes	N/A
	OSM	228	yes	ODbL
Friuli	Asso Rifugi	21	yes	N/A
	OSM	54	yes	ODbL
	Alpi FVG	33	no	N/A

Table 3.2: Huts from the Italian Alps regions

Surprisingly, the richest source in terms of number of elements is OSM, which is also the only global source among the ones in Table 3.2. Also, OSM was the most precise among them, as a few of the local sources gave frequent incorrect coordinates or coordinates only relative to the area of hut, but precision is fundamental for PeakLens. Another problem was the copyright of these sources: only a few of them specified their terms of usage, while others should have been directly contacted in order to understand if it was possible to use their data. Another point in favor of OSM is the API that it provides: no one of the local sources found had any API endpoint. As a consequence, OSM was elected as the principal data source.

The second global data source, analyzed for improving the data about peaks,

is WikiData. This source, compared to OSM, provides fewer elements, but its contribution is relevant from the names point of view: PeakLens is an international App and, as such, it needs to provide meaningful peaks' names depending on the nationality of the user because, as it is possible to see from [Figure 3.4](#), the names may be very diverse.

Name	Languages	Name	Languages	Name	Languages
Mont Blanc	en, fr, es, de, af, ast, bar, bs, ca, cs, cy, da, et, eu, fi, fo, frp, fy, gl, hr, hu, id, ie, ku, lb, nl, nn, oc, pl, rm, ro, sk, sl, sv, sw, tr, vls, nb, jv, sh, war, sco, ceb, ilo	مون بلان	fa, mzn	Blianc Mont	nrm
		מוֹן בְּלַן	he	ماؤنت بلائک	pnb
		Մոնբլան	hy	Monte Branco	pt
		மொண்டர் பிளாங்கு	ta	Munti Jancu	scn
		Hvitfjall	is	Mont Blank	sq
	モモンブブララン	ja	Monto Blanka	io	
Monte Bianco	it, vec	მონბლანი	ka, xmf	ຍອດເຂາມງບລັອງ	th
ሞን ብላን	am	မြန်မာ	km	مونت بلائک	ur
مون بلون	ar	몽몽블블랑 산산	ko	Đinh Blanc	vi
Манблан	be	Rupes Alba	la	勃勃朗朗峰峰	zh
Монблан	bg, kk, os, ru, sr, tg, tt, uk, ba, ky	Mont Bianch	lmo	מאן בלאנק	yi
		Monblanas	lt	Menez Gwenn	br
মোঁ ব্লাঁ	bn	Monblāns	lv	माँट ब्लाँक	mr
Λευκό Όρος	el	Мон Блан	mk	白白朗朗峰峰	yue
Blanka Monto	eo	मोन्ट ब्लाङ्क	ne	Monblan	az, uz

Figure 3.4: Monte Bianco Names

WikiData provided a good amount of alternative names for peaks, which were not present in the OSM database. All the elements present in WikiData are identified by a code. This WikiData code is also present in most of the OSM's elements, making it possible to have a direct correspondence between two elements that are the same. Following this procedure of names integration almost 120 thousands names were added to PeakLens.

Earth Models data sources

A crucial data for AR representation of geographic elements is the elevation which is used to place the labels precisely, but it is also shown to the user as an important information about the element. Elevations are often missing from OSM (almost 40000 peaks higher than 1000m without reported elevation) or sometimes wrong in terms of unit of measure (e.g. elevation specified in feet instead of meters).

An alternative source that is possible to use to establish the elevation of a certain element is the Digital Elevation Model (DEM) of the terrain. The most popular DEM, which does not cover poles and can contain errors and

holes, has been released by the NASA's SRTM project [4].

Two types of SRTM DEM are publicly available at the time of writing: DEM 1 arcsecond, which has a resolution of 30m at the equator, and DEM 3 arcseconds, which has a resolution of 90m at the equator. Obviously DEM 1 is more precise, but the amount of data to manage both on a server and on the client would largely increase, possibly introducing performance problems and worsening the user experience.

At the time of writing, the PeakLens App uses the DEM 1 resolution only for the Alps area and the DEM 3 for the rest of the world. In particular, the version of the DEM used in the implementation of this work is not the original SRTM's DEM, but a version improved over the years from a user and obtainable from ViewFinder Panoramas [16]. This version of the DEM is based on SRTM data [4], but also uses many other local data sources to integrate and improve the data of the original one, solving a good part of the holes and missing data problem.

3.1.3 POI importing process

This section describes the approach developed to import and store a variety of data from a global data source. This process is divided in four main steps:

- The execution of queries to retrieve specific typologies of data
- The integration of other metadata from a secondary source
 - Alternative names from WikiData
 - Elevations from DEM
- The automated check to filter out errors from the source
- The storage of data in a structure that allows one to selectively provide these data to the end user based on their location

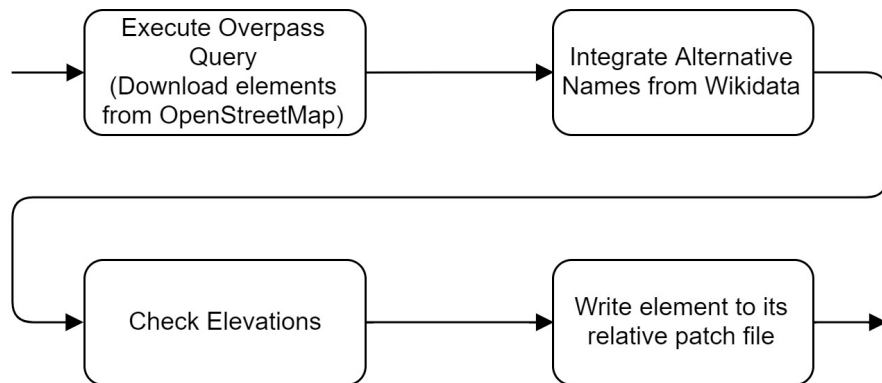


Figure 3.5: The steps of importing POIs in PeakLens

Importing OSM data at scale

Importing a great amount of geo-referenced data at global scale can be a difficult task; updating such data frequently, to keep them in sync with user’s edits in the original source, can be even more complex. One of the options to solve this problem is to download all the available data and keep only the ones that are needed for a given purpose.

The solution proposed in this work implies the use of the Overpass API introduced in [subsection 1.1.2](#) that allows one to create a local replica of the data by querying the OSM original database specifying a variety of filters. Furthermore, the API offers an integrated system to keep the local copy of the data updated, by synchronizing them with the main OSM origin server.

The creation of a synchronized local replica of the OSM data can significantly improve the performance of the local data processing pipeline and the quality and timeliness of the data used in the mobile AR app.

The process of importing data in the PeakLens local replica begins with a query to the OSM Overpass API, requesting a specific type of elements (e.g. peaks) from all around the world. The only conditions necessary for the elements not to be discarded is that they must have a name and the coordinates. The stream of data received from the Overpass API contains all the requested elements whose names are integrated with the ones retrieved from Wikidata.

Correcting OSM elevations of objects with elevations inferred from the DEM

The DEM allowed us to improve the quality of elevation data acquired from the OSM local replica, in this way:

- The elevation used in order to place the label in the correct position is the one extracted from the DEM in the element's location.
- The elevation written on the label as an info for the user is the one saved in OSM, if it is not present the label is showed anyway, but without elevation.
- If the difference between the elevations from OSM and the DEM is greater than a certain threshold the elevation from OSM is considered wrong and it will not be shown to the end user.

As stated above, there are many cases in which the elevations retrieved from OSM are considered wrong, this may depend on different factors:

Referring to [Figure 3.6](#):

1. Typing error
2. Approximate elevations (e.g. "ca 1400" or "~1400")
3. The elevation is expressed in a unity of measure different from meters
4. The elevation extracted from the DEM and the one from OSM are both correct, but the coordinates specified in OSM are wrong and, as a consequence, the elevation from those coordinates is not the one of the real element. This is a problem that the algorithm explained in [section 3.3](#) aims at resolving in the future, detecting peaks from OSM that are outside a detected peak area.

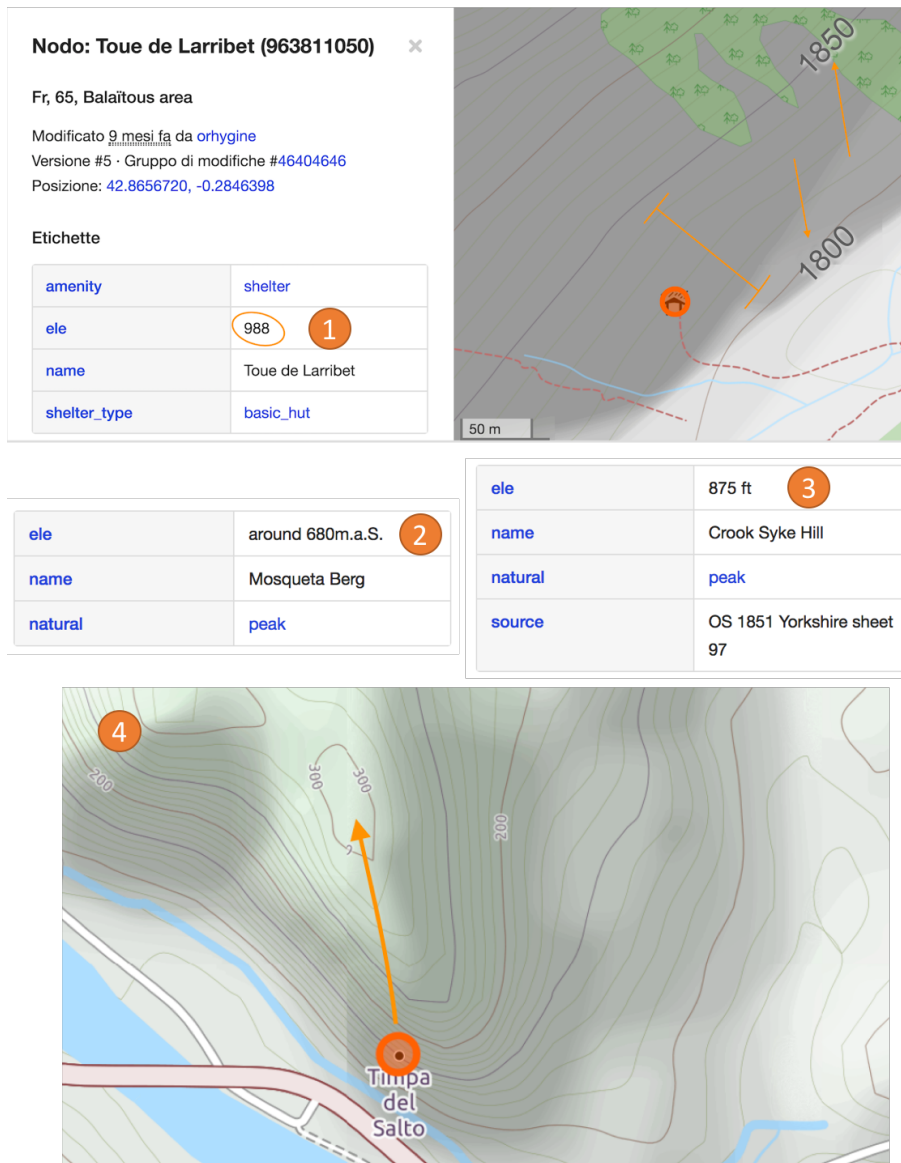


Figure 3.6: Elevation errors from OSM GUI

The process of correcting the elevations is illustrated in Figure 3.7. If an elevation is undefined, no checking is necessary. Otherwise, the comparison between OSM elevation and DEM elevation starts: at first the difference is calculated and, if it is below a threshold, the OSM elevation is accepted and stored. If the difference is above the threshold a conversion from feet to meters is applied and the result is compared to the DEM elevation again. At this point, if the threshold is respected the data are saved or else discarded.

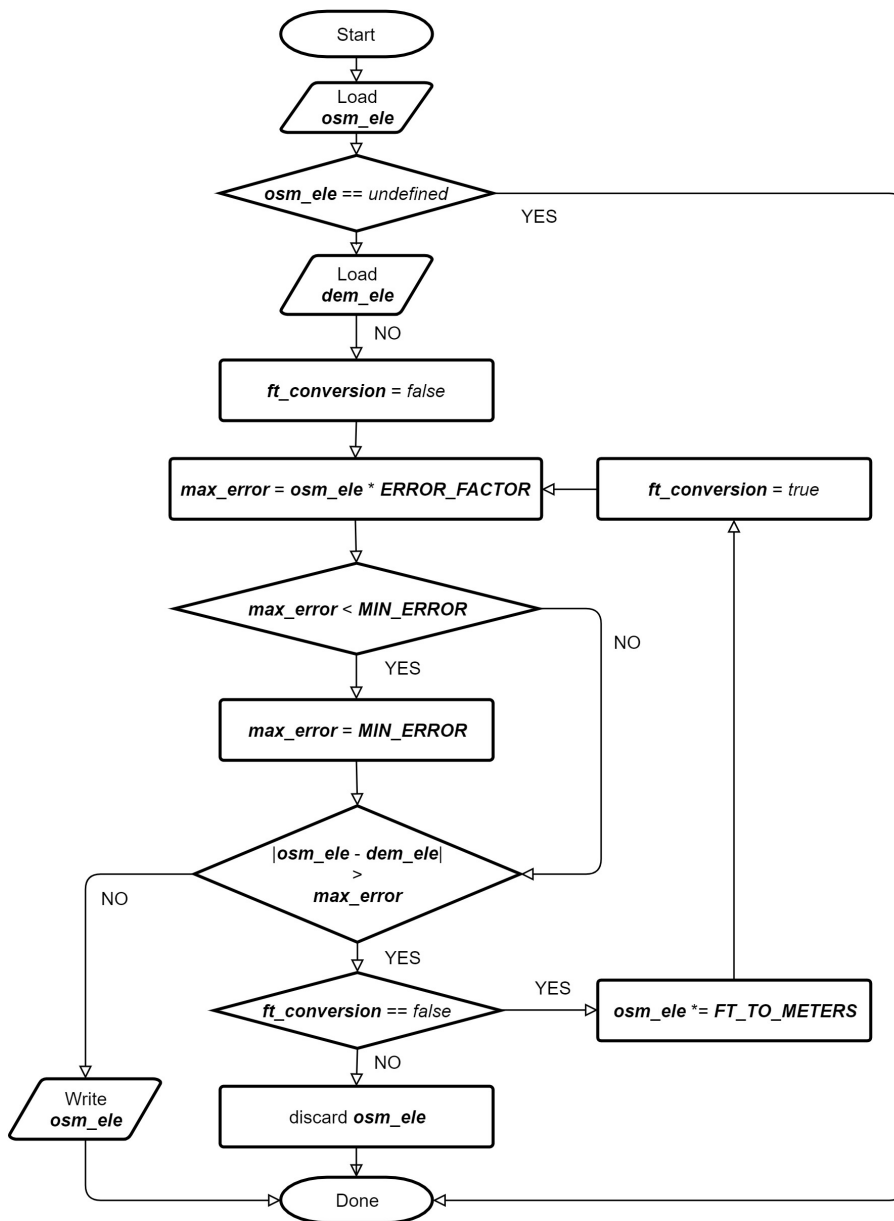


Figure 3.7: Elevation Check Process

During the development of this algorithm it was noticed that it was not possible to use an absolute threshold to decide whether a user inserted elevation is an error or not: to an higher real elevation of the element corresponds, in fact, a greater error in respect to the one obtained from the DEM (see Table 3.3). As a consequence, the threshold used in the process to accept or discard the OSM elevation is a relative one, i.e., proportional to the OSM

elevation. For example, if a peak’s elevation from OSM is very high, let’s say 7000m, an elevation from the DEM different by 200m will be accepted, but it will not be accepted for a peak 1000m high. A minimum absolute threshold exists because of the presence of small peaks, which would make the proportional threshold too restrictive.

Range [m]	Average difference [m]
1000-2000	28
2000-3000	44
3000-4000	65
4000-5000	100
5000-6000	166
6000-7000	280
7000-8000	271

Table 3.3: Elevation difference between DEM and OSM

At this point the downloaded data are ready to be saved in the PeakLens format that will be explained extensively in the next chapter

Data storage

The data organization in the PeakLens server must cope with the fact that the user has to be served with only the data that they need for a specific location, identified by the GPS sensor of the phone. To provide the portion of data around the user efficiently, it was decided to organize metadata extracted from OSM under the same structure used for the DEM data. The data format is called HGT (contraction for height) and it was created to store Nasa’s SRTM data files [4].

An HGT file covers an area of $1^\circ \times 1^\circ$: for example, the file N20E100.hgt contains data from 20°N to 21°N and from 100°E to 101°E exclusive. This format seemed suitable to store all the elements separately following the same structure: peaks and huts will be contained in a folder with the same name as the DEM cell, a name that represents the coordinate limits in which they are located. In this way, when the user connects to the PeakLens server, they provide their position and the server, loading the data around the user, provides a Virtual Panorama from their point of view.

3.2 Augmented reality rendering of monodimensional elements

The POI rendering module explained in this section was developed within the PeakLens mobile application, that is, as already explained in [subsection 1.1.1](#), an application able to label a panorama of peak areas seen from the smart phone camera.

The contribution of the thesis is to extend the currently visualization module of PeakLens to handle the rendering of different types of elements.

The architecture follows the client-server model: on the server the Digital Elevation Model of the entire planet is stored together with all the elements to be visualized, the client makes requests to the server exposing its location in order to retrieve the virtual panorama and the visible elements.

In the following subsections all the steps of the visualization process, embodied in the extended visualization module, are explained.

3.2.1 Identification of position and orientation

The first step is the sensor activation and initialization. This is automatically done at the launch of the camera functionality. To work properly the orientation sensors needs to be calibrated, so a message that invites the user to move the smartphone around in an eight patterned figure is shown ([Figure 3.8](#)).



Figure 3.8: Orientation Sensors Calibration: how to move the device to help the sensor calibration.

The movement forces the orientation sensors to reset, restart and return the correct cardinal direction. The user is also invited to activate the GPS by means of a pop-up. Data on position are mandatory for the following phases, so it is not possible to use the application without activating it. After the calibration and GPS activation steps, the interface that will host all the elements above the images captured by the camera is shown [26].

3.2.2 Virtual Panorama and Visibility

Once the application is started and all the sensors activated and calibrated, a request is made to the server in order to get all the data needed to display contents to the user in the right way.

Using the algorithm created for [22], the Digital Elevation Model of the area surrounding the position sent by the user is loaded and used to create a virtual panorama. This is achieved by projecting on a cylinder the elevation data and shading all the pixels using the distance between that point and the user.

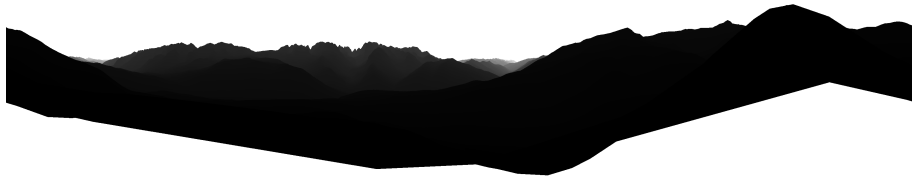


Figure 3.9: Virtual Panorama

At the same time also all the elements in a given radius from the user are loaded and projected in the 3D space. This procedure also considers the language of the user in order to provide understandable results. Since the user's language is not always available in the dataset, the following policy is applied: if the name of an element is available in the language of the user, this one is used, otherwise, if present, English is used leaving the original language only as the last chance.

With all these information the system is able to define for each element whether it is visible from the user point of view or not by comparing the distance of the element from the user and the distance of its projection on

the virtual panorama.

In that way only the visible elements are returned to the user together with the skyline extracted from the virtual panorama.

3.2.3 Pagination

In mountain areas, that is the main field of use of PeakLens, many elements like peaks and huts are to be displayed in a space constrained by the screen size of the smartphone. In order to display all the elements in a readable way also in outdoor conditions, PeakLens already uses a method called *pagination*. The goal is to distribute peaks in different pages in order to avoid overlapping between their labels. In this section is explained this method and how to modify it to be applied also to other elements.

```
Data:  $x_{MIN}$  // Min distance between two peaks
        peaks // List of peaks
Result: Every peak in peaks is assigned to a page
1 pages = [] // Empty list that will be filled of pages
2 peaks = peaks sorted by y coordinate
3 foreach peak1 in peaks do
4     assigned = false
5     foreach page in pages do
6         ok = true
7         foreach peak2 assigned to page do
8             if  $|x_{peak1} - x_{peak2}| < x_{MIN}$  then
9                 ok = false
10                break
11            end
12        end
13        if ok then
14            assign peak1 to page
15            assigned = true
16            break
17        end
18    end
19    if !assigned then
20        add a new page to pages and assign peak1 to it
21    end
22 end
```

Algorithm 3.1: Peaks Pagination

Pagination is achieved, as formalized in [algorithm 3.1](#), considering peaks sorted by their vertical position in the virtual panorama and adding them, one by one, to the first page able to contain it. A peak can be assigned to a page if and only if it is far enough from all the others peaks already present in that page. This distance is computed looking only at the x coordinate of the peaks projected on the virtual panorama.

With the addition of new elements outside of peaks, the algorithm explained immediately above is no longer suitable for handling all the cases. In fact huts are often lower than peaks and the probability to go in the last pages is very high even if, laying at two different elevations, they may also be placed in the same page without overlapping. [Table 3.4](#) reports an example in which peaks and huts are treated as the same type of element during pagination. The result is that 9 out of 10 huts are allocated after the fifth page.

Page	N° Peaks	N° Huts
1	17	1
2	17	0
3	16	0
4	16	0
5	16	0
>5	862	10

Table 3.4: Result of the pagination when peaks and huts are treated as the same type of element

This led to the development of a multiple steps pagination algorithm composed by three operations:

1. Pagination of peaks (see [algorithm 3.1](#))
2. Creation of a set of bindings between huts and pages considering the already positioned peaks (see [algorithm 3.2](#))
3. Pagination of huts following the bindings created at point 2 (see [algorithm 3.3](#))

```

Data:  $x_{MIN}$  // Minimum horizontal distance
          $y_{MIN}$  // Minimum vertical distance
Result: Set of bindings
1 foreach page of peak-pages do
2   foreach h of huts do
3     foreach p of peaks in page page do
4       if  $|x_h - x_p| < x_{MIN}$  and  $(|y_h - y_p| < y_{MIN}$  or  $y_p < y_h)$  then
5         create a binding for h in page
6       end
7     end
8   end
9 end

```

Algorithm 3.2: Bindings Creation

Peaks are the most important, and also numerous, elements of the interface; it is imperative that the addition of other elements does not interfere with their rendition. For this reason the pagination of peaks is performed as the first step. In this way it is impossible that a peak is moved to another page due to the presence of another element.

For this reason, huts cannot be placed freely in any page but they are subjected to some bindings computed with [algorithm 3.2](#).

In order to prevent overlapping, the creation of bindings compares, page by page, all the huts with every peak in that page. The condition is that every time a peak and a hut are close on the x axis, the hut must be under the peak and the vertical distance must be greater than a threshold; otherwise a binding is created. In particular, a hut can be placed in a page if and only if the following formula holds for every peak in that page

$$|x_h - x_p| \geq x_{MIN} \quad \vee \quad (|y_h - y_p| \geq y_{MIN} \quad \wedge \quad y_p \geq y_h)$$

where:

x_h, y_h are the coordinates of a hut in the virtual panorama

x_p, y_p are the coordinates of a peak in the virtual panorama

x_{MIN} is the minimum horizontal distance

y_{MIN} is the minimum vertical distance

```

Data:  $x_{MIN}$  // Min distance between two huts
        huts // List of huts
        bindings // Set of bindings
Result: Every hut in huts is assigned to a page
1 pages = [] // Empty list that will be filled of pages
2 huts = huts sorted by distance from the user
3 foreach hut1 in huts do
4     assigned = false
5     foreach page in pages do
6         if hut1 has not a bindings with page then
7             ok = true
8             foreach hut2 assigned to page do
9                 if  $|x_{hut1} - x_{hut2}| < x_{MIN}$  then
10                    ok = false
11                    break
12                end
13            end
14            if ok then
15                assign hut1 to page
16                assigned = true
17                break
18            end
19        end
20    end
21    while !assigned do
22        add a new page to pages if hut1 has not a bindings with page then
23            assign hut1 to page assigned = true
24        end
25    end
26 end

```

Algorithm 3.3: Huts Pagination

Once bindings are created, also huts can be paginated following the procedure reported in [algorithm 3.3](#) that is very similar to the one of peaks. In this case huts are sorted by distance in order to give greater importance to the elements closer to the user. Using this ordering, the algorithm try to add each hut to the pages where it has no binding and finally add the element to a page if it is at least x_{MIN} distant on the x axis from all the other huts in that page.

Page	N° Peaks	N° Huts
1	18	5
2	17	3
3	16	1
4	16	1
5	16	1
>5	861	0

Table 3.5: Result of the pagination done with the multiple steps algorithm with bindings creation

Performing this three steps procedure on the same example presented in [Table 3.4](#), the result reported in [Table 3.5](#) is achieved. In this case all the huts are paginated in the first five pages leaving also the space for one more peak.

3.2.4 Alignment of object positions with the camera view

All the operations explained so far, which involved only the sensors and the virtual panorama, were done to produce a distribution of elements visible from the point of view of the user on various pages, to guarantee visualization without overlapping.

Another fundamental operation is the alignment of the virtual panorama with the image captured by the camera in order to place all the elements in the right position. This is achieved thanks to an algorithm that has been developed in the SnowWatch and PeakLens projects, explained in [subsection 1.1.1](#). The complete explanation of the algorithm can be found in [\[22, 26\]](#). In the following, we provide only a brief introduction.

In order to position the image on the virtual panorama, the image is rescaled computing its field of view from the focal length and the size of the camera sensor and considering that the virtual panorama has a field of view of 360° .

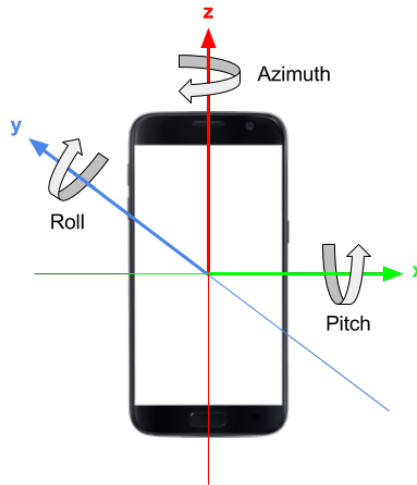


Figure 3.10: Sensors data. Figure taken from [26]

The first step is the identification of elements positions by only using sensors data which make available the following values (see Figure 3.10):

azimuth angle between the device’s current compass direction and magnetic north

pitch angle between a plane perpendicular to the device’s screen and a plane parallel to the ground

roll angle between a plane perpendicular to the device’s screen and a plane perpendicular to the ground

In this phase the orientation of the phone in term of azimuth, pitch and roll is used to estimate the position of the image acquired by the camera on the virtual panorama [26].

In order to refine this initial estimation that might contain imprecisions due to sensor errors, a comparison between the mountain skyline extracted from the image and the one extracted from the virtual panorama is performed.

The skyline extraction is based on the application of a Convolutional Neural Network supervised learning algorithm, which finds the landscape skyline, i.e., the set of all points that represent the boundary between terrain slopes and the sky [22]. This method is capable of distinguishing the skyline from

other elements of the photo such as trees or clouds, even in case of partial occlusions caused by objects such as buildings [26].

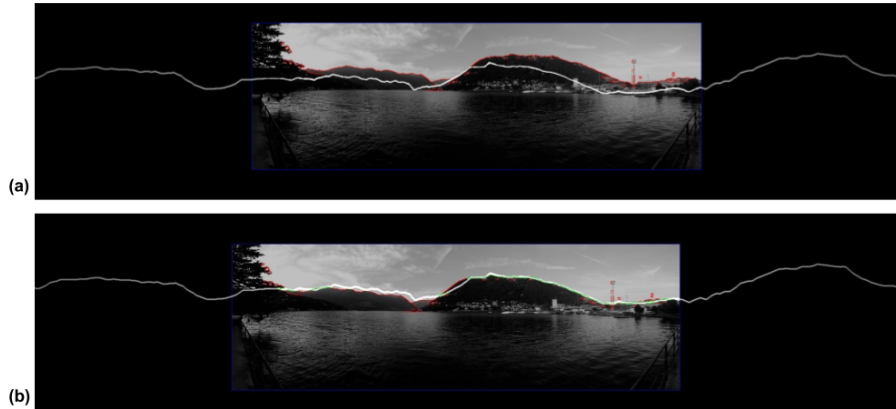


Figure 3.11: Comparison between sensor and global matching taken from [26]:
 (a) panorama aligned with the reference skyline by using sensor data
 (b) alignment refined after the global matching algorithm.

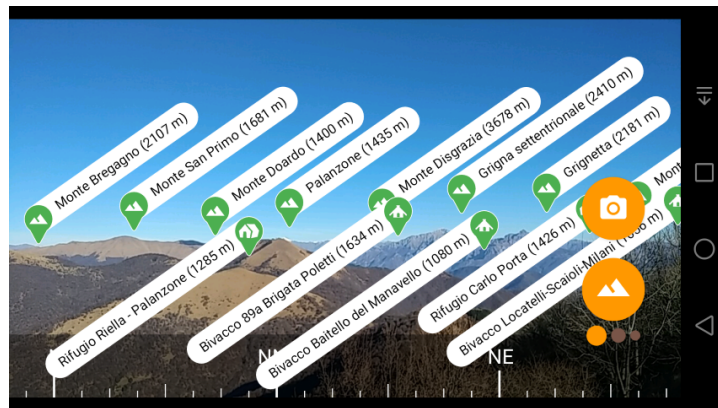
Given the position computed by sensors and the two skylines the **global matching algorithm** is applied to find the best position of the image with respect to the virtual panorama. The best position is considered to be the one in which the 2 skylines overlap more. This is found by applying some vertical and horizontal displacements in a given interval and computing a score for each position. The score is based on the degree of overlap of the two skylines. Finally, the position with the highest score is returned as an offset with respect to the initial position and it is added to the elements positions. Positions computed in this step should be closer to the actual peak positions than the ones computed by only using sensor data. Figure 3.11 shows a comparison between (a) the alignment using sensor data only and (b) the result after the application of the global matching algorithm. The positioning in (b) is clearly better. In both figures the skyline extracted from the panorama is highlighted in red and in (b) the green color has been used to highlight pieces that overlap with the reference skyline [26].

Only with regard to the peaks, also a **local alignment** is applied in order to improve their precision. For each peak it is considered a local neighborhood centered in the photograph location identified as the peak position by the global alignment. In this way each peak position is refined by identifying the best match in its local neighborhood [22]. The local alignment technique cannot be applied as is to elements that are not on the skyline because their

presence can be detected neither by the virtual panorama nor by analyzing the image. In future it would be possible to take for each hut some peaks of reference and move them accordingly.

3.2.5 Object Visualization in the Camera View

While pagination is done only once for a given user's position, the alignment is continuously executed, trying to process as much camera frames as possible to create a near real time update of the view following the user and device movement.



(a) Labels with opaque white background



(b) Semitransparent labels with different size for secondary elements

Figure 3.12: Main interface of PeakLens showing labels before (a) and after (b) the restyling

Every time a frame is processed, the objects have to be displayed on top of the image coming from the camera using the positions found during the alignment and the grouping in pages created in the pagination phase.

The first problem faced is the positioning of labels. As already discussed during pagination in [subsection 3.2.3](#), an important point was that new elements must not interfere with peaks. Peaks have a slanted label that starts from the peak position facing upwards. Maintaining this style for peaks, it was decided to use labels with the same inclination, but that point downwards, in order not to invade the space of peaks.

In designing the visualization of the elements, it was very important to remember that all the names must be clearly readable also in outdoor conditions. This led, in the version of PeakLens containing only peaks, to the use of labels with a opaque white background. This type of visualization that is very nice for peaks, resulted not suitable for other elements since peaks occupies only the top part of the screen, while other elements also the bottom one, covering too much of the panorama.

In order to alleviate the unpleasant effect of masking the panorama, it was decided to reduce the labels' background opacity and border for non peaks elements. In this way, the panorama is less occluded maintaining a good level of readability. In addition to that, if more elements will be added in future, a filter that allow the user to select the elements in which they are interested, will be useful. At the moment this filter is not present in order to leave the application as simple as possible for the user.

3.3 Automatic peaks extraction from DEM data

PeakLens is used by more than 120k users worldwide; some of them reported in their reviews that not always the dataset included all the peaks in the area they were visiting.

One of the aim of this work is to improve the quality of data and so also to increase the amount of information at disposal.

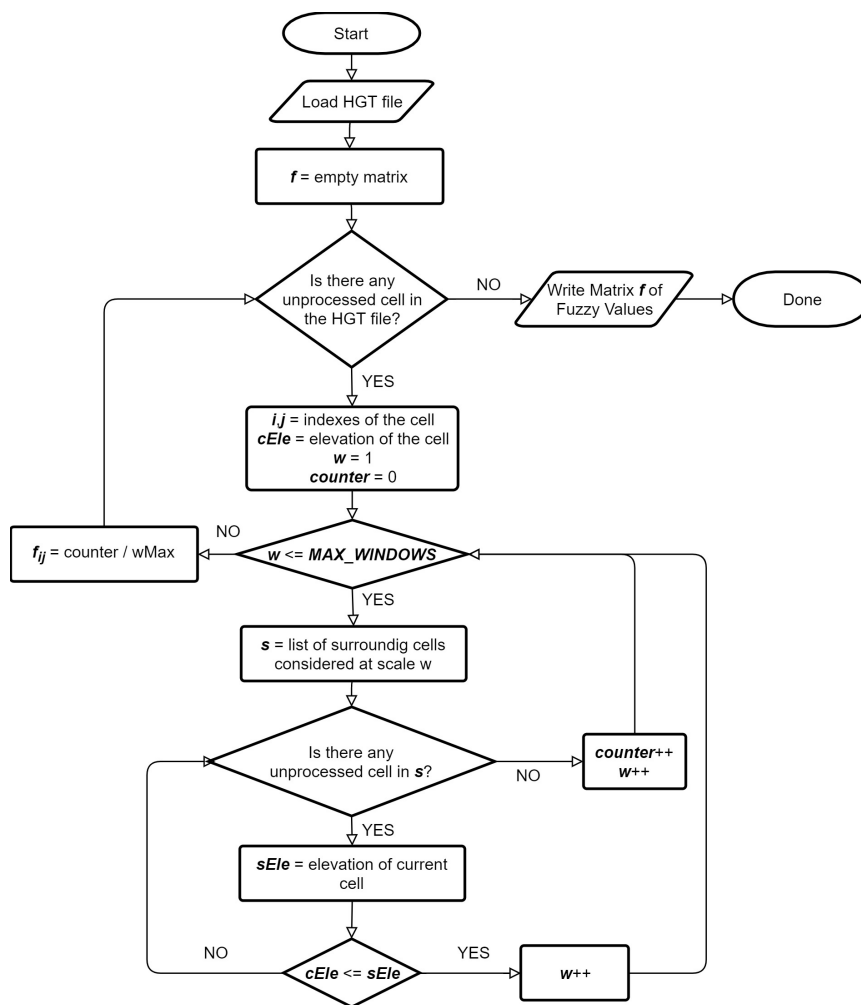


Figure 3.13: Flowchart diagram of peaks extraction defined in [23]

To increase the number of peaks available to the application, beyond the limitations of the OSM dataset, the idea came to use the Digital Elevation

Model to discover new peaks, by analyzing the morphology of the land. In order to do that, we started studying and testing the approaches to geomorphological analysis, e.g., as explained in [section 2.2](#). In particular we focused on peaks and ridges; as a first experiment we implemented the peak detection method explained in [\[23\]](#).

This algorithm can be characterized as a multi-scale method that directly extracts peaks areas from the Digital Elevation Model. The procedure is described in [Figure 3.13](#): it takes as input an HGT file representing the DEM and produces a matrix of the confidence values of the presence of a peak within a cell. The only parameter is the maximum number of windows that is the number of scales to be analyzed.

Right from the beginning this method has shown quite interesting results that immediately led to the development of a small tool to visualize the results directly on a map taken from OpenStreetMap. This tool allowed to go through the single cases and discover the problems present in this method together with the possible improvements. The main problems are the following:

- difficulties in recognizing series of mountains ranged in a line (see [Figure 5.3\(a\)](#))
- difficulties in detecting standalone peaks not in mountain areas (see [Figure 5.4\(a\)](#))
- lack of control over the prominence of the peaks to extract.

The solutions to these problems, explained in the following sections, have led to the formulation of an improved algorithm, composed by the following steps:

- find ridge areas
- filter ridge areas by size
- find peak areas in ridge areas
- filter peak areas by fuzzy threshold
- estimate peak position within each peak area

Figure 3.14 shows a graphical representation of the flow of data.

This algorithm considers different scales both for ridges and peaks computation but using different window size; therefore it can be considered a multi-scale approach. On the other hand it is not a pure multi-method approach because it does not compute peaks in two different ways but uses two methods in order to constraint the discovery of peaks only in the areas of interest allowing to perform a different analysis.

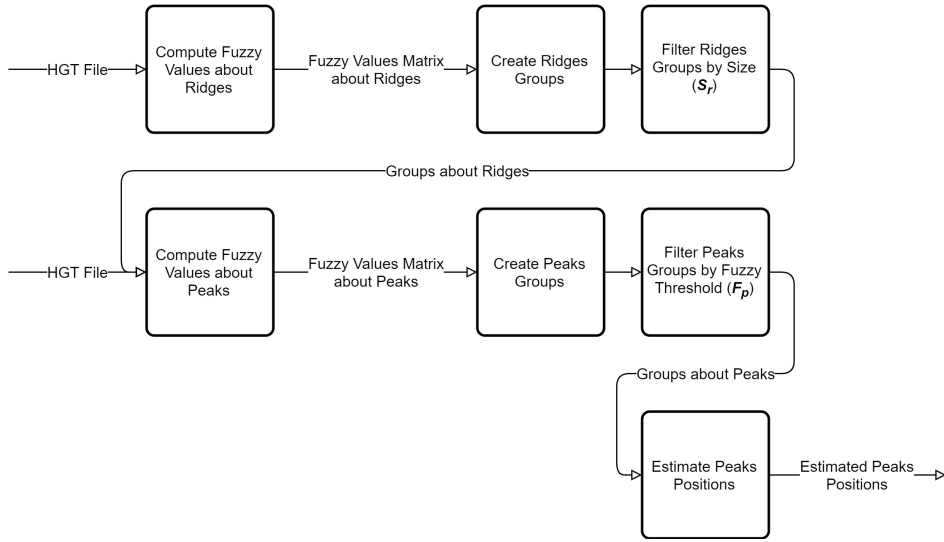


Figure 3.14: Overall process designed to find peaks. Rectangles represent computation phases and edges the flow of data.

The following parameters characterize the whole process and its output:

P_r Elevation gap during cells comparison for ridges computation

W_r Maximum number of windows / scales to be analyzed for ridges computation

F_r Minimum level of confidence for a cell to be considered part of a ridge

S_r Minimum size of a ridge in number of cells

P_p Elevation gap during cells comparison for peaks computation

W_p Maximum number of windows / scales to be analyzed for peaks computation

F_p Minimum level of confidence for a group of cells in order to assert the presence of a peak

3.3.1 Detection of Ridges

In order to focus the research of peaks only in mountain areas, it was decided to preliminary find ridges. Ridges are the lines that intersect the two sides of a mountain and as such they represent the areas where peaks lie.

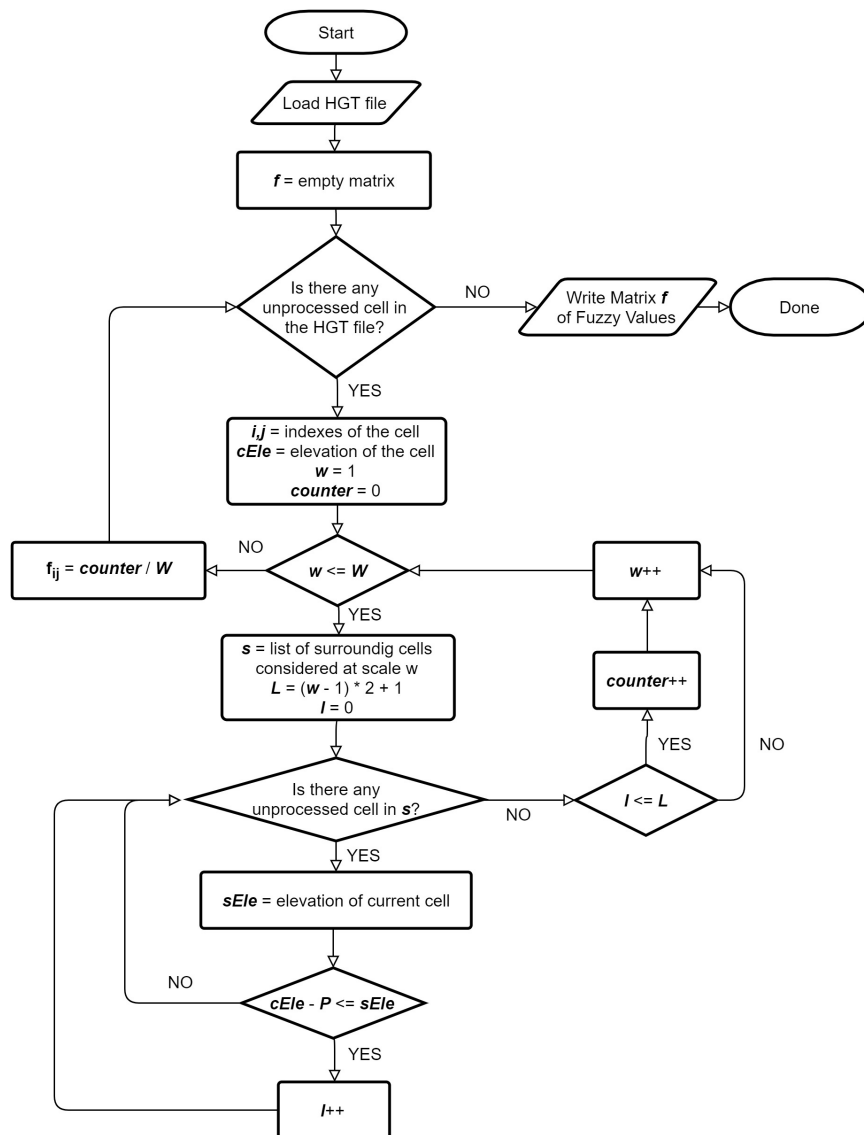


Figure 3.15: Flowchart diagram of ridges extraction

The ridges extraction process is represented in [Figure 3.15](#), which is a modification of [Figure 3.13](#).

Considering the Digital Elevation Model as a grid, what has been done is to scroll this grid cell by cell and for each one compute a fuzzy value that is the confidence of a cell to be on a ridge. The fuzzy value is computed analyzing the surrounding area at different scales and making an average of the results. At each scale a different set of cells is considered by taking the neighbors of the cell under analysis at the first step and the external neighbors of the previous set at the next steps so that a cell is considered for the analysis of only one scale (see [Figure 3.16](#)). This technique has been already mentioned as a moving window of increasing size.

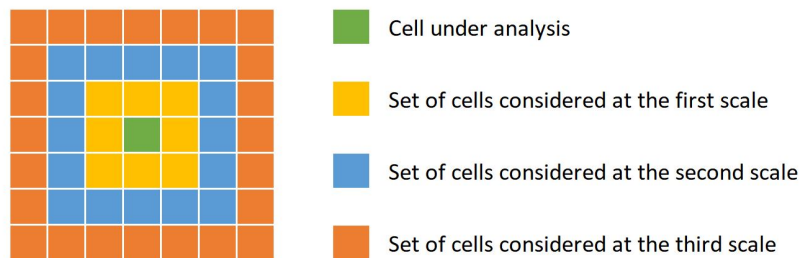


Figure 3.16: Representation of sets of cells, which form the increasing window, considered during the computation of the fuzzy value. In this example 3 scales are considered that creates windows of 3x3, 5x5 and 7x7.

The outcome of the analysis performed at each scale is a Boolean value that specifies whether a cell is part of a ridge or not; this value is translated into 0 or 1 in order to compute the average between the results achieved at the other scales.

The Boolean value computed at each scale is the effect of the comparison between the elevation of the cell under analysis and the elevations of the cells considered at that scale. In particular a cell is marked with true at a specific scale if, comparing its elevation with the others, it is P_r meters higher than at least L cells belonging to the current set where L is dynamically computed for each scale in a way that is always equals to the number of cells that composes one side of the square minus 2. The introduction of P_r with respect to [23] allows to recognize only quite prominent ridges.

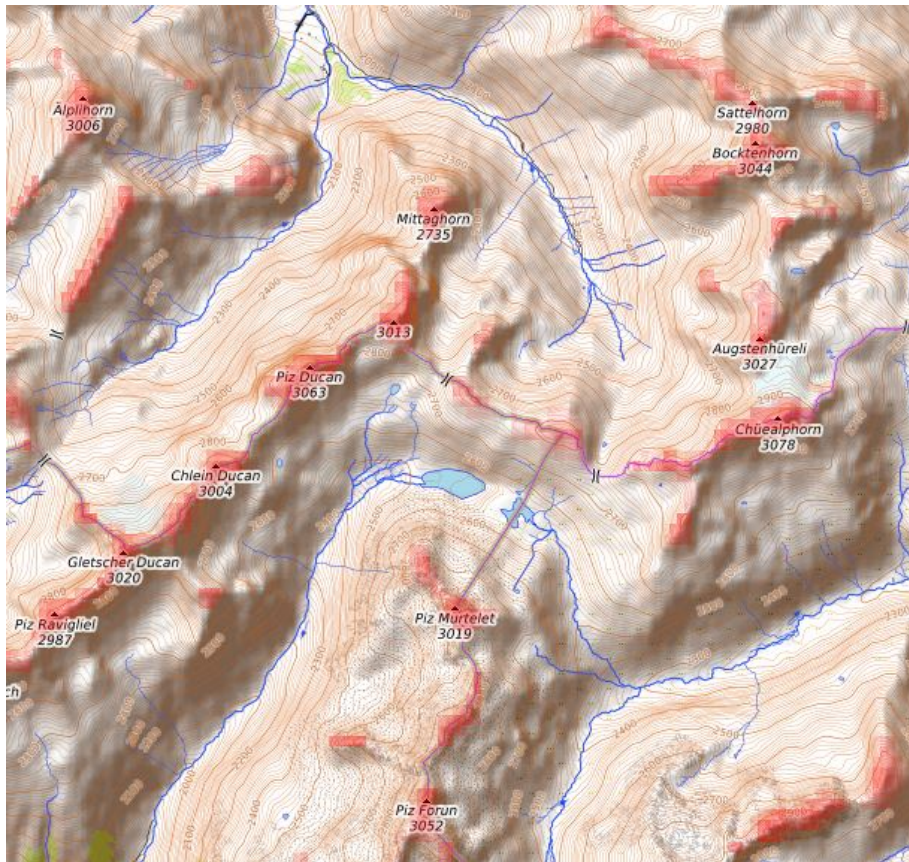


Figure 3.17: Representation on a map of fuzzy values denoting ridges

The overall result is a matrix of values between 0 and 1 that indicate the confidence for a cell to be on a ridge. In Figure 3.17 fuzzy values are represented as various shades of red over a map.

Cells with a fuzzy value higher than F_r are grouped with their neighbors. This is done following the procedure described in Figure 3.18.

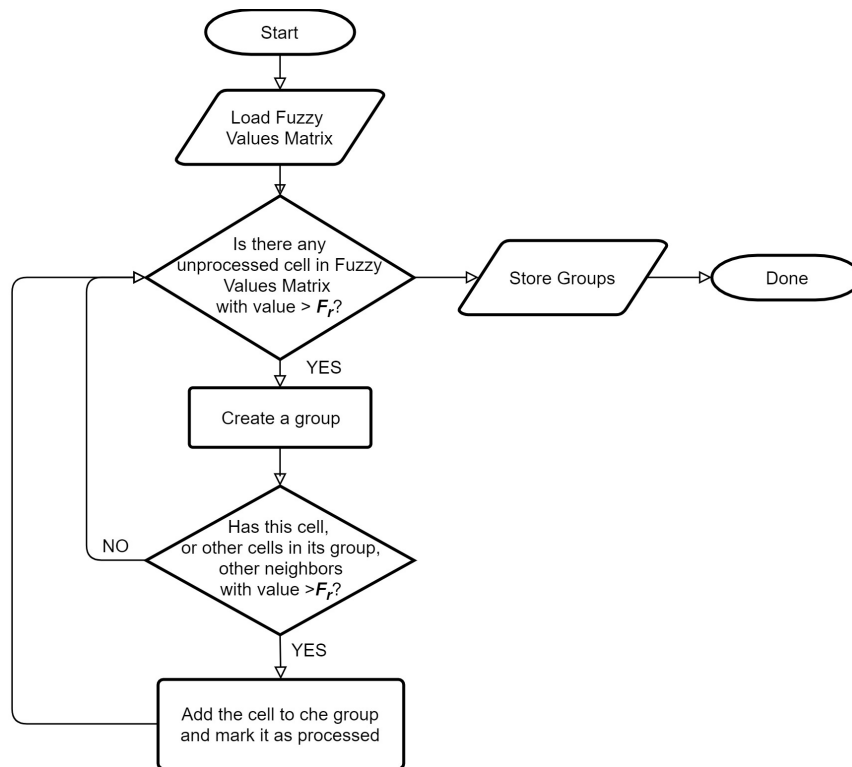


Figure 3.18: Flowchart diagram of groups creation

These groups are filtered by size removing all those smaller than S_r .

3.3.2 Detection of Peaks

For each cell that has been found to belong to a ridge following the previous procedure, it is necessary to check whether it contains a peak or not.

The procedure is quite similar and also easier. The flowchart in [Figure 3.15](#) is valid also for this task with the following modifications:

- L has not to be computed but always be 0
- Only the cells that have been found in ridge computation are considered (instead of the entire DEM). Consequently, the need to load also the groups found in ridge computation arises.

The first step is to compute the fuzzy value: for each scale a Boolean value

saying if the cell contains a peak or not is computed and the values coming from different scales are averaged together.

In order to state if a cell contains a peak at a given scale, its elevation must be P_p meters higher than all the cells belonging to the set of surrounding cells considered at that scale. P_p , a fixed threshold introduced at the beginning of this section, is used to set how much prominent a peak must be to get recognized.

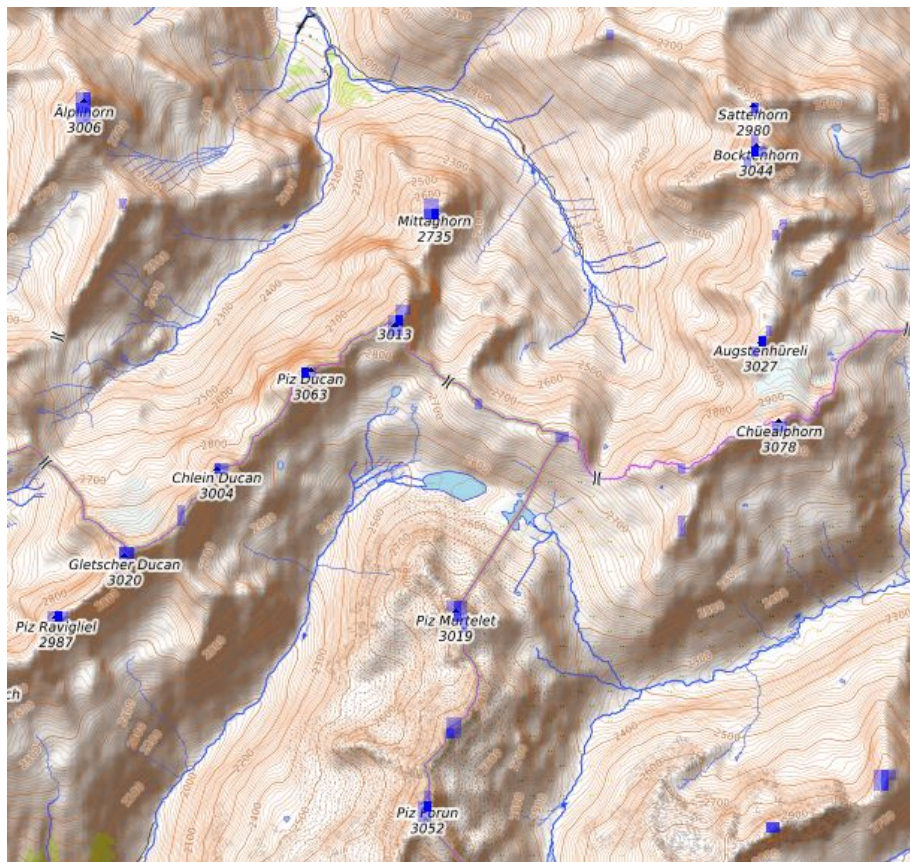


Figure 3.19: Representation on a map of fuzzy values denoting peaks

Also in this case the result is a matrix of values between 0 and 1 representing the confidence value of the presence of a peak within a cell.

Due to the fuzziness of values there is not a one-to-one correspondence between a peak and a cell but often a peak is discovered by the presence of more adjoining cells with a strictly positive value. For this reason cells close to each other are grouped together and the group has its own fuzzy value:

the sum of the values of the cells included in that group. Based on this value, groups are filtered in order to remove the less significant ones.

The remaining groups represent the peaks that have been discovered during this process. In order to estimate the position of each peak, a weighted average is used: in particular the estimated latitude (longitude) of the peak is calculated as the average of the latitude (longitude) of the center of each cell belonging to the group weighted using the fuzzy value of each cell.

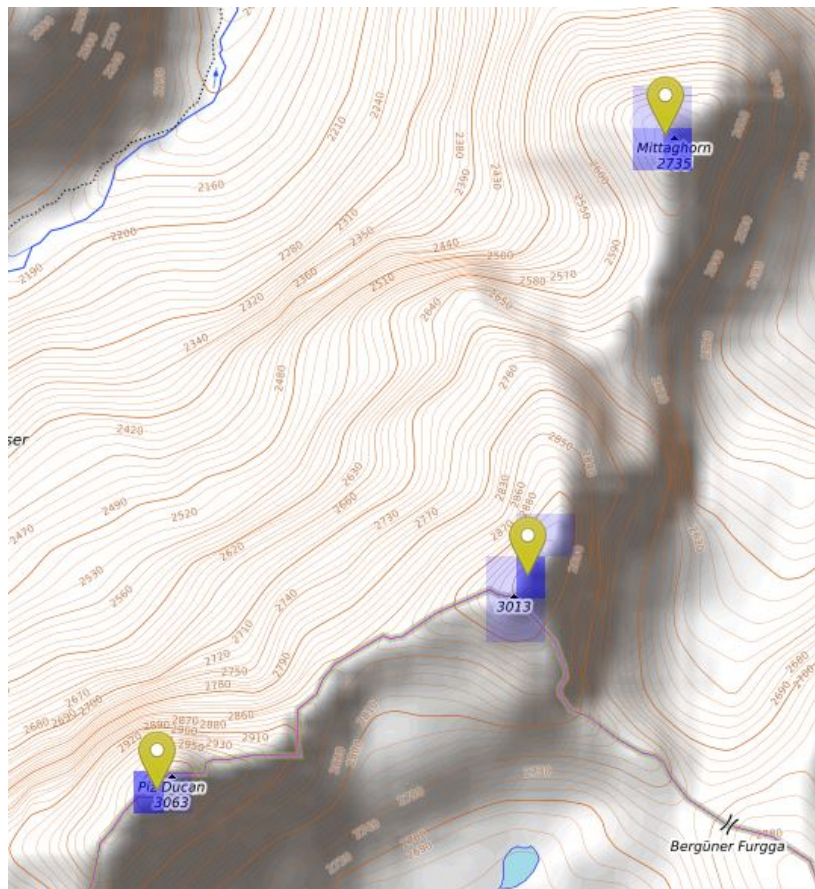


Figure 3.20: Peaks positioning within groups of cells

Chapter 4

Implementation Details

This chapter presents the implementation details of the various modules developed or extended by this work.

Figure 4.1 shows a general overview of the components that constitute the data processing and visualization of the PeakLens case study.

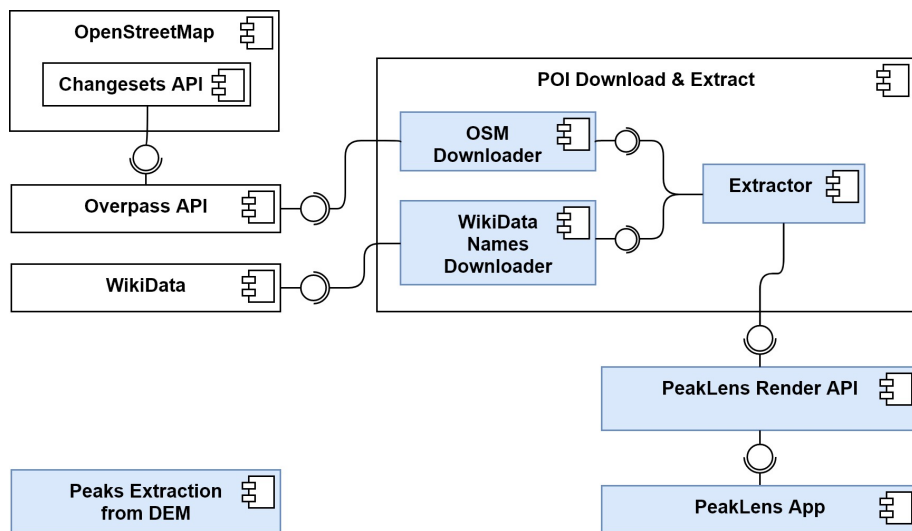


Figure 4.1: Main Components of the system

The *POI Download and Extract* module represents the part of Data Retrieval used to create the PeakLens dataset. This part is written in *Javascript* programming language running in a *Node.js* environment.

The *PeakLens Render API* module is responsible for the generation of the Virtual Panorama and the collection of all the visible elements; it constitutes the server side of the PeakLens application. The programming languages used in this part are *C++* and *Javascript*. The *C++* code is compiled through *node-gyp* and using *bindings* and *nan* is possible to call *C++* functions from *Javascript* running in a *Node.js* environment. The *express* framework is used to handle http requests from the client.

The *PeakLens App* is the Android application able to display peaks in augmented reality that, at the end of this work, also acquires the capability to show other elements. The programming language used is *Java*. Its Computer Vision module, already present, uses also the *OpenCV* library.

The *Peaks Extraction from DEM* module is used to discover new peaks analyzing the Digital Elevation Model. This module uses *Javascript* as programming language.

4.1 Data retrieval

This section describes the details about the specific interaction with the OSM and WikiData API and the implementation of a script to import all the elements that will constitute the PeakLens App dataset.

4.1.1 OpenStreetMap and Overpass API

The Overpass API is publicly available and usable by everyone. The main problem about the public instances is that they are hosted by third party companies/institutions and there is no guarantee that they are always on-line or that accept big queries.

From the first tests it has been noticed that the time of response was not regular and sometimes timeouts happened making it necessary to repeat large queries from scratch. As a consequence, a local instance of Overpass was installed on PoliCloud in order to have an always available service to keep PeakLens data updated. In fact, an Overpass instance is natively able to update the data in its database directly from OSM. This is one of the main feature that led to the choice of using the Overpass API, as it is very important to let the user update data on OSM and see the results on PeakLens as soon as possible. In this case the installed instance updates every hour.

OpenStreetMap base elements

Understanding how the data are provided by OSM is fundamental to proceed to the download and storage. Nodes, ways and relations are the three type of base elements present in OSM.

- a node represents a specific point on the earth's surface defined by its latitude and longitude. Each node comprises at least an id number and a pair of coordinates. A node can be included as member of a way or a relation. The relation also may indicate the member's role: that is, the node's function in this particular set of related data elements.[7]
- a way is an ordered list of between 2 and 2,000 nodes that define a polyline. Ways are used to represent linear features such as rivers

and roads. Ways can also represent the boundaries of areas (solid polygons) such as buildings or forests. In this case, the way's first and last node will be the same. This is called a "closed way". [7]

- a relation is a multi-purpose data structure that documents a relationship between two or more data elements (nodes, ways, and/or other relations). The relation is primarily an ordered list of nodes, ways, or other relations. These objects are known as the relation's members. A single element such as a particular way may appear multiple times in a relation. [7]

While peaks are always represented by a node, other elements like huts are represented by way or relations used to mark the limits of the building, which cannot be assimilated to a single point. Listing 4.1 is an example of a hut element from OSM: it is noticeable that this element is represented by a way which, in turn, comprehends many nodes.

```

1 <? xml version="1.0" encoding=" UTF-8"?>
2 < osm version="0.6" generator=" CGImap 0.6.0 (23262 thorn-01.
  openstreetmap. org)" copyright=" OpenStreetMap and
  contributors" attribution=" http:// www. openstreetmap.
  org/ copyright" license=" http:// opendatacommons. org/
  licenses/ odbl/1-0/">
3   < way id="84679331" visible=" true" version="6"
  changeset="53317882" timestamp="2017-10-28 T13:25:51 Z"
  user=" Creando" uid="132740">
4     < nd ref="983649757" />
5     < nd ref="983649760" />
6     < nd ref="983649764" />
7     < nd ref="983649765" />
8     < nd ref="983649757" />
9     < tag k=" amenity" v=" restaurant" />
10    < tag k=" building" v=" yes" />
11    < tag k=" cuisine" v=" alpine_hut" />
12    < tag k=" ele" v="2438" />
13    < tag k=" name" v=" Dreizinnenhutte - Rifugio
  Locatelli alle Tre Cime di Lavaredo" />
14    < tag k=" name: de" v=" Dreizinnenhutte" />
15    < tag k=" name: it" v=" Rifugio Locatelli alle Tre
  Cime di Lavaredo" />
16    < tag k=" tourism" v=" alpine_hut" />
17    < tag k=" website" v=" http:// www. dreizinnenhutte.
  com" />
18    < tag k=" wikidata" v=" Q1258258" />
19    < tag k=" wikipedia" v=" de: Dreizinnenhutte" />
20  </ way>
21 </ osm>

```

Listing 4.1: Example of hut element from OSM in XML format

Querying the data

As stated in [subsection 4.1.1](#) huts and buildings in general are not characterized by a single point in OSM. Therefore, before the inclusion of huts it was possible to avoid using an Overpass server by downloading the whole OSM globe containing all the elements in OSM database and, using a stream approach, store only the elements representing peaks. In order to import huts (and other elements in future developments) it is not possible to use a simple stream approach to keep only specific elements while downloading. Info about this nodes, such as coordinates and elevation, are not reported directly in this element, but the nodes id has to be queried directly. The downloaded globe has the same structure and there is no order for nodes in respect to ways and relations, as a consequence it is not possible to store data while downloading without storing anything in memory. An Overpass server, thanks to the database it relies on, offers the possibility to retrieve data about relations and ways returning also the data about the single nodes they comprehend. At the same time it has to be considered that answering such a query may take several minutes and keeping the connection open while waiting for the result should be avoided. [Listing 4.2](#) is an example of a possible query to the OverPass server that asks to retrieve all the basic elements of "alpine hut" type, which is a sub-category of tourism. The only filter imposed by this query is the fact that the elements must have a name.

```
1 [ out: json][ timeout:7200];
2 (
3   node[" tourism"=" alpine_hut"][" name"];
4   way[" tourism"=" alpine_hut"][" name"];
5   relation[" tourism"=" alpine_hut"][" name"];
6 );
7 out geom meta;
```

Listing 4.2: Query to retrieve all the alpine hut elements from the Overpass instance

In this implementation the *Express* framework installed on the proprietary server receives the request containing the query and forward the latter to the running overpass instance which will compute the result, while the response to the client will contain an id relative to the performed query. When the query result is ready, it is saved locally. Since the time the client receives the response id relative to the query, it repeatedly sends a request to the Overpass server in order to know if the query result is ready. When the result is ready the download to the client starts. This implementation was

chosen to avoid losing the query result because of connection faults: if the connection is lost the client repeats the request and the result is still present on the server, without executing the query again. An example of response is presented in [Listing 4.3](#).

```
1 {
2   " generator" : " Overpass  API",
3   " elements" : [{
4     " type" : " way",
5     " id" : 84679331,
6     " timestamp" : " 2017- 10- 28T13: 25: 51Z",
7     " version" : 6,
8     " changeset" : 53317882,
9     " user" : " Creando",
10    " uid" : 132740,
11    " bounds" : {
12      " minlat" : 46.6368285,
13      " minlon" : 12.3103324,
14      " maxlat" : 46.6370235,
15      " maxlon" : 12.3107023
16    },
17    " nodes" : [ 983649757, 983649760, 983649764,
18                 983649765, 983649757],
19    " geometry" : [
20      { " lat" : 46.6368285, " lon" : 12.3106366 },
21      { " lat" : 46.6369168, " lon" : 12.3103324 },
22      { " lat" : 46.6370235, " lon" : 12.3103981 },
23      { " lat" : 46.6369352, " lon" : 12.3107023 },
24      { " lat" : 46.6368285, " lon" : 12.3106366 }
25    ],
26    " tags" : {
27      " amenity" : " restaurant",
28      " building" : " yes",
29      " cuisine" : " alpine_hut",
30      " ele" : " 2438",
31      " name" : " Dreizinnenhütte - Rifugio Locatelli alle
32                Tre Cime di Lavaredo",
33      " name: de" : " Dreizinnenhütte",
34      " name: it" : " Rifugio Locatelli alle Tre Cime di
35                Lavaredo",
36      " tourism" : " alpine_hut",
37      " website" : " http:// www.dreizinnenhuette.com",
38      " wikidata" : " Q1258258",
39      " wikipedia" : " de: Dreizinnenhütte"
40    }
41  }
42 }
```

Listing 4.3: Overpass response example

4.1.2 WikiData

WikiData provides an endpoint which allows to query multiple ids in order to get all the information stored in their database.

```
1 {
2   " entities" :{
3     " Q583" :{
4       " type" : " item",
5       " id" : " Q583",
6       " labels" :{
7         " en" :{
8           " language" : " en",
9           " value" : " Mont Blanc"
10        },
11        " it" :{
12          " language" : " it",
13          " value" : " Monte Bianco"
14        },
15        " es" :{
16          " language" : " es",
17          " value" : " Mont Blanc"
18        },
19        " de" :{
20          " language" : " de",
21          " value" : " Mont Blanc"
22        },
23        " af" :{
24          " language" : " af",
25          " value" : " Mont Blanc"
26        },
27        " br" :{
28          " language" : " br",
29          " value" : " Menez Gwenn"
30        },
31        " pt" :{
32          " language" : " pt",
33          " value" : " Monte Branco"
34        }
35      }
36    }
37  }
38 }
```

Listing 4.4: Example response from WikiData

As it is possible to notice from [Listing 4.4](#) a *JSON* response comprehending

all the names present in WikiData is provided. Relative to the "labels" key the available names are identified by their language prefix (e.g. "it","en"...), these prefixes are the same as the ones used in OSM, making it easy to integrate these names if they are missing from OSM data.

4.1.3 Stream approach

One of the main problems with managing these data is that their growth is constant and it is not possible to completely rely on the server RAM to store the data while elaborating them. The stream approach allows the process to be of low impact on the memory independently of the size of the streamed data. Working with streaming data means that these data are processed incrementally without having access to all of them at the same time.

In this work the response given by the Overpass server is treated as a stream of elements and, as a consequence, during the import process only one element at a time is loaded and processed. The main libraries used to work with streams are *JSONStream* [3] and *EventStream* [2]. These libraries are used both to download and convert the data from the Overpass server and to generate the PeakLens data structure.

Listing 4.5 presents an example of *GeoJSON* file. *GeoJSON* is a geospatial data interchange format based on *JavaScript Object Notation (JSON)*. It defines several types of *JSON* objects and the manner in which they are combined to represent data about geographic features, their properties, and their spatial extents. *GeoJSON* uses a geographic coordinate reference system, *World Geodetic System 1984*, and units of decimal degrees[11]. The *GeoJSON* standard format has been chosen to become the intermediary format between a general source data format and the PeakLens data format. In this way the data acquisition process is completely decoupled from the successive phases.

```

1 {
2   " type" : " FeatureCollection",
3   " features" : [
4     {
5       " type" : " Feature",
6       " properties" : {
7         " @id" : " way/ 84679331",
8         " amenity" : " restaurant",
9         " building" : " yes",
10        " cuisine" : " alpine_hut",
11        " ele" : " 2438",
12        " name" : " Dreizinnenhütte - Rifugio Locatelli alle
13          Tre Cime di Lavaredo",
14        " name: de" : " Dreizinnenhütte",
15        " name: it" : " Rifugio Locatelli alle Tre Cime di
16          Lavaredo",
17        " tourism" : " alpine_hut",
18        " website" : " http:// www.dreizinnenhuetten.com",
19        " wikidata" : " Q1258258",
20        " wikipedia" : " de: Dreizinnenhütte",
21        " @timestamp" : " 2017- 10- 28T13: 25: 51Z",
22        " @version" : 6,
23        " @changeset" : 53317882,
24        " @user" : " Creando",
25        " @uid" : 132740
26      },
27      " geometry" : {
28        " type" : " Polygon",
29        " coordinates" : [
30          [ 12.3106366, 46.6368285],
31          [ 12.3107023, 46.6369352],
32          [ 12.3103981, 46.6370235],
33          [ 12.3103324, 46.6369168],
34          [ 12.3106366, 46.6368285]
35        ]
36      },
37      " id" : " way/ 84679331"
38    }
39  ]
40 }

```

Listing 4.5: GeoJSON Example

Listing 4.6 shows how the *JSONStream* library allows to work with a JSON file as if it was a stream of keys and values: a *GeoJSON* file is opened and streamed through the `JSONStream.parse` function, which allows to emit

from the stream only the elements specified by a given pattern specified as parameter. *EventStream* allows to treat a stream as an array, which is not in memory, but distributed in time. This makes it easier to manage the flow of data emitted from the stream and "map" their structure to the PeakLens data structure.

```
1 function getFeaturesStream( fileName) {
2   let stream = fs. createReadStream( fileName, {
3     encoding: ' utf8'
4   });
5   return stream. pipe( JSONStream. parse(' features.*' ));
6 }
```

Listing 4.6: JSON Stream usage example

4.1.4 PeakLens Data Structure

[Listing 4.7](#) shows the structure of an element saved in the PeakLens server: the main features of this structure are the coordinate field (**coord**), which can store from one to many coordinates, making it possible to store multi-coordinates elements, and the **ref** field, which is used to link a PeakLens element to the respective element from another source.

```
1 {
2   " id" : " way/ 84679331",
3   " coord" : {
4     " lat" : 46.6369065,
5     " lon" : 12.3105412,
6     " ele" : 2403
7   },
8   " display_ele" : 2438,
9   " name" : " Dreizinnenhütte - Rifugio Locatelli alle Tre
10  Cime di Lavaredo",
11  " alt_name" : {
12    " de" : " Dreizinnenhütte",
13    " it" : " Rifugio Locatelli alle Tre Cime di
14    Lavaredo",
15    " nl" : " Dreizinnenhütte"
16  },
17  " ref" : {
18    " wiki" : " Q1258258"
19  },
20  " ts" : " 2016- 07- 01T15: 51: 40Z"
21 }
```

Listing 4.7: PeakLens data format

4.2 Augmented reality rendering of monodimensional elements

This section describes the technical details relative to rendering in augmented reality of peaks and huts.

In [Figure 4.2](#) is presented the architecture that is used in this part.

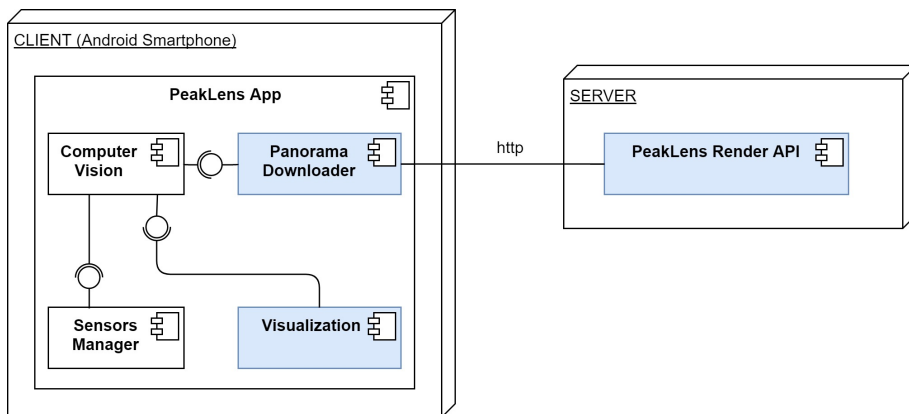


Figure 4.2: PeakLens Client-Server Infrastructure

The architecture is composed by a client and a server. The server is responsible of the generation of the virtual panorama when requested by the client and return it together with all the elements to be displayed in augmented reality. The client, that is an Android Smartphone equipped with the PeakLens application, has to perform all the other operations such as paginate elements, handling sensors, aligning the panorama and show the labels in augmented reality.

4.2.1 PeakLens Render API

The Render API runs server side and is responsible of the generation of the virtual panorama together with all the visible elements from the point specified in the request made by the client.

In [Figure 4.3](#) the classes' schema is presented. The panorama class is not real but only a placeholder used to represent the part of the system already present able to create the virtual panorama, extract the skyline and compute

elevations from the Digital Elevation Model.

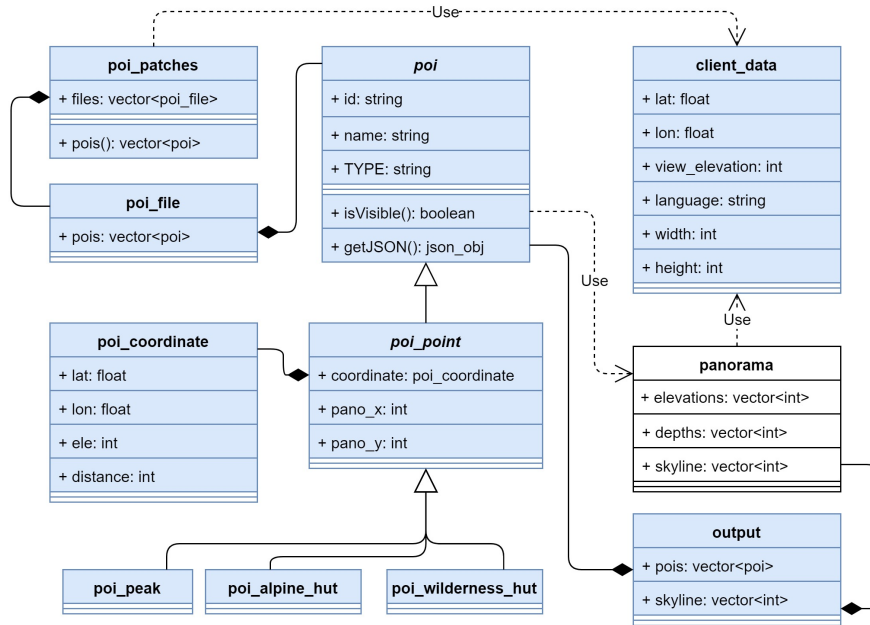


Figure 4.3: PeakLens Render API Class Diagram

The client performs a **HTTP GET** specifying its position in term of latitude and longitude, the preferred language, and the dimension of the virtual panorama. These information are stored in **client_data** enriching them with the computed elevation relative to the position of the user (**view_elevation**).

The virtual panorama is immediately computed using the parameters specified in the request. The result is a matrix of dimensions specified by the client and flattened row by row in the **depths** array that provide the distance of each point of the virtual panorama from the user. The **skyline** array has length equals to the width of the virtual panorama and store the height of the skyline at each point.

Then, all the elements to be displayed in a fixed radius from the user are loaded following the **poi** hierarchy represented in Figure 4.3 and projected on the virtual panorama (**pano_x**, **pano_y**). These elements must be filtered in order to get only the visible elements, namely, all the elements that have not any mountain between them and the user. This is done comparing the linear distance between the element and the user and the distance taken from the corresponding point in the **depths** array.

```

1 {
2   " width" : 3600,
3   " height" : 1800,
4   " elevation" : 2064,
5   " distance" : 115387.5546875,
6   " skyline" : [ 769, 770, 771, 771, 772, 772, 773,
773, 774, 774, 774, 775, 775, 775, 775,
775, 775, 775, 776, 776, 776, 776, 776,
776, 776, 777, 777, 777, 777, 778, 778,
... ],
7   " poi" : [
8     {
9       " id" : " node/ 26862689",
10      " name" : " Cima Brenta",
11      " position" : {
12        " distance" : 9184,
13        " ele" : 3112,
14        " x" : 1242,
15        " y" : 835
16      },
17      " type" : " PEAK"
18    }, {
19      " display_ele" : 2182,
20      " id" : " way/ 38612573",
21      " name" : " Rifugio Maria e Alberto al Brentei",
22      " position" : {
23        " distance" : 8076,
24        " ele" : 2170,
25        " x" : 1345,
26        " y" : 893
27      },
28      " type" : " ALPINE_HUT"
29    }, {
30      " id" : " node/ 1388378223",
31      " name" : " Bivacco Costanzi",
32      " position" : {
33        " distance" : 9340,
34        " ele" : 2365,
35        " x" : 510,
36        " y" : 882
37      },
38      " type" : " WILDERNESS_HUT"
39    }
40  ]
}

```

Listing 4.8: PeakLens Render API Response

At this point all the data are collected in a *JSON* object to be returned to the client. An example is presented in [Listing 4.8](#). In particular the **skyline**

array is included and an array with all the visible elements is generated taking their *JSON* representation calling the `getJSON()` function on each element.

4.2.2 Virtual Panorama Downloader

The next phases are performed client side on an Android Smartphone, so the *Java* programming language is used.

Once PeakLens is started and the GPS position is acquired, a request is made to the previously explained service. The same type of request is performed also when the GPS registers a new position or a more precise one. An example of response is reported in [Listing 4.8](#).

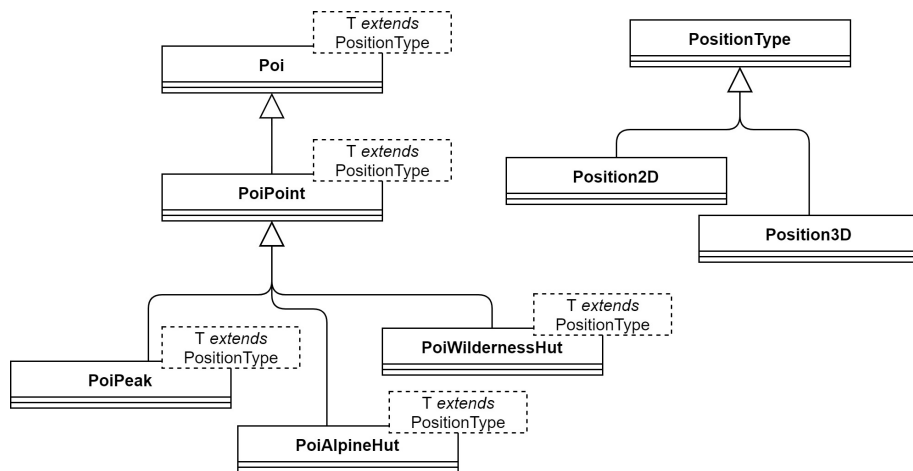


Figure 4.4: PeakLens App Pois Class Diagram

Parsing the *JSON* response, a Panorama representation containing all the elements and the skyline is created. The elements are stored following a hierarchy similar to the one used server side and is represented in [Figure 4.4](#). The adoption of *Generics* allowed to use the same classes to represent the elements both in the 3D panorama and in the 2D one by applying one of the two `PositionType`. This distinction is needed by the Computer Vision module used to align the Virtual Panorama on the camera image.

Once the panorama representation is ready, the pagination must be per-

formed following the algorithms explained in [subsection 3.2.3](#).

At this point the Computer Vision module can be started.

4.2.3 Visualization

The visualization phase has to combine two types of data: the image coming from the camera and the elements to draw on top of. In order to do that it is very important the contribution of the Computer Vision module that allows to know the position of each element relative to the image taken from the camera.

Each frame acquired from the camera is sent to the Computer Vision to be processed but, due to the computational complexity of the Computer Vision module, not all the frames will be processed. In particular the Computer Vision starts to compute a new frame only when the previous one is completed and all the frames received in the meantime are discarded.

Every time the Computer Vision module ends the computation of a frame, all the elements are redrawn on the screen in order to adjust their position.

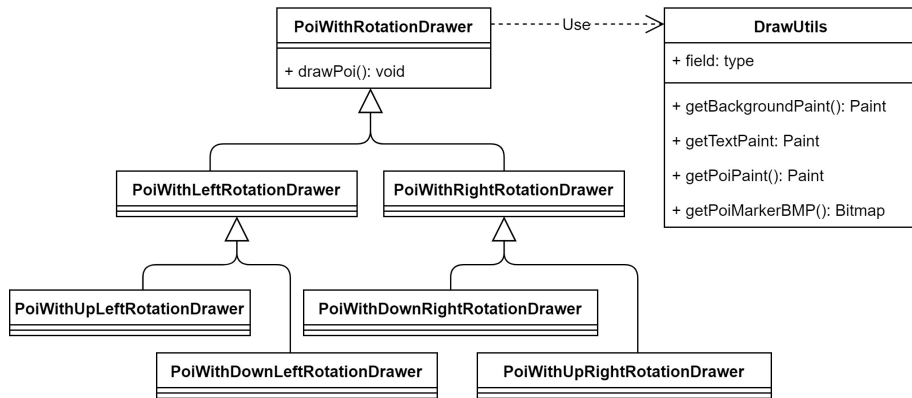


Figure 4.5: PeakLens App Drawers Class Diagram

In order to draw labels on top of the image, a transparent *view* with a *canvas* is placed on the image. The different styles are obtained by setting the correct drawer from the hierarchy presented in [Figure 4.5](#). In particular the left or right choice is taken looking at the *roll* of the device retrieved from the sensors (see [Figure 3.10](#)) while the up or down style is selected looking at the element type (see [Listing 4.9](#)).

```
1 public class PoiWithLeftRotationDrawer implements
  PoiDrawer {
2
3     private static final PoiWithLeftRotationDrawer
      INSTANCE = new PoiWithLeftRotationDrawer();
4
5     public static PoiWithLeftRotationDrawer getInstance(){
6         return INSTANCE;
7     }
8
9     public void drawPoi( Canvas canvas, Poi< Position2D>
      poi, boolean isPicture){
10         if ( poi. getPoiType() == PoiType. PEAK){
11             PoiWithUpLeftRotationDrawer. getInstance().
              drawPoi( canvas, poi, isPicture);
12         } else {
13             PoiWithDownLeftRotationDrawer. getInstance().
              drawPoi( canvas, poi, isPicture);
14         }
15     }
16
17 }
```

Listing 4.9: Poi Drawer

4.3 Peaks extraction from DEM

Following the ideas explained in [section 3.3](#), here will follow the most important details about the algorithm used to extract peaks from the Digital Elevation Models.

4.3.1 HGT files

In order to explain the whole process, is necessary to clearly understand in which form the DEM is available: the HGT files.

A HGT file covers a surface of 1 degree x 1 degree and contains a number of values that depends on its resolution. It is stored as a binary file in "raw" format (no headers and not compressed) and values are stored as 16-bit signed integers one after the other as to form an array [5]. Nevertheless, the Digital Elevation Model is always been considered as a grid: in order to connect the two representations it is enough to say that the array is derived by flattening the grid row by row. The only parameter needed to perform this transformation is the size of the grid that is strictly related to the resolution of the model. An important note is that, given the resolution, the grid is always one row and one column larger: this is very important during the transformation but these values can be discarded because they overlap with the adjacent files.

In order to locate the grid on the earth, HGT files have a name which identifies the coordinate of the bottom left corner of the grid.

This type of partitioning, namely the storage of data in patches of 1 degree x 1 degree, is used also for all other files generated during the process. This is very useful in order to always load in memory a small amount of data and keep the process as modular as possible.

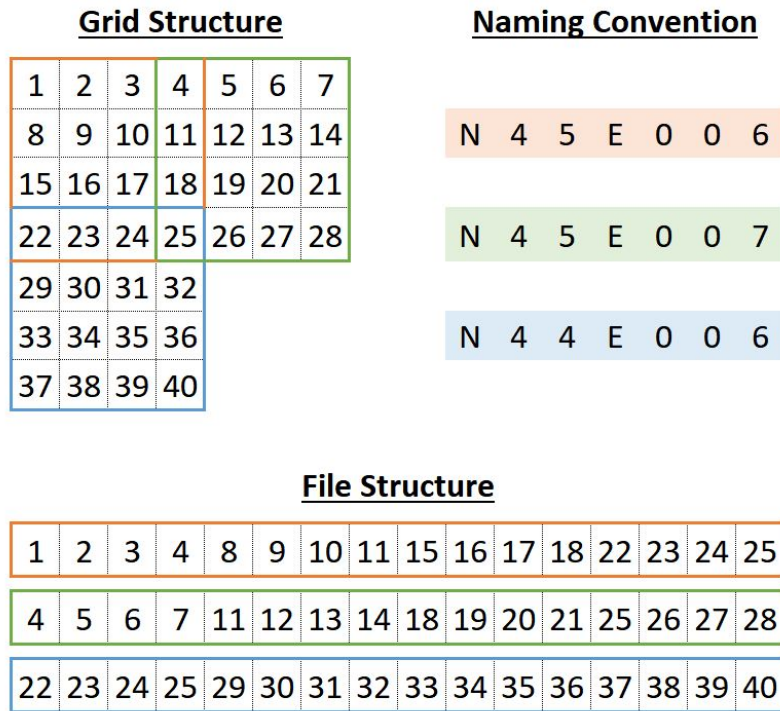


Figure 4.6: HGT Structures Example: Here are used dummy HGT files of 4 rows and 4 columns as an example. It makes clear the parallelism between the file structure (how the HGT files are stored on disk) and the grid structure that is the usual way to think about them. Also an example about the naming convention is added.

One drawback of this approach is the following: if one patch is loaded and a cell at its border is being analyzed, it is also necessary to load the adjacent patch in order to complete the analysis with all the surrounding cells. The solution adopted to solve this problem is to always load the patch under analysis and the other eight around it. The coordinates (rows and columns) to access the grid are always referenced to the central patch in order to have direct access to it and, when they go out of the indexes of that patch, after locating the landing patch, they are translated into its own coordinate system.

4.3.2 Fuzzy Values Computation

The heart of the problem is the computation of fuzzy values used to understand if a cell belongs to a ridge or contains a peak.

The method used for ridges and peaks is almost the same and in this section it is explained jointly.

```
1 function computeCellFuzzyValue( multiHgt, size, row, col,
2   maxWindow, steepness, withLoss) {
3   let readFunction;
4   if ( row > maxWindow && row < size - maxWindow && col >
5     maxWindow && col < size - maxWindow)
6     readFunction = function( i, j) {
7       return readHGTByIndex( multiHgt. c, size, i, j);
8     }
9   else
10    readFunction = function( i, j) {
11      return readMultiHgtByIndex( multiHgt, size, i, j);
12    }
13
14  let centerElevation = readFunction( row, col);
15  let counter = 0;
16  for ( w = 1; w <= maxWindow; w++) {
17    const maxLoss = withLoss ? ( w - 1) * 2 + 1 : 0;
18    let loss = 0;
19    for ( j of [ col - w, col + w]) {
20      for ( i = row - w; loss <= maxLoss && i <= row +
21        w; i++) {
22        if ( centerElevation - steepness <= readFunction( i,
23          j))
24          loss++;
25        }
26      }
27    }
28    for ( i of [ row - w, row + w]) {
29      for ( j = col - w + 1; loss <= maxLoss && j <=
30        col + w - 1; j++) {
31        if ( centerElevation - steepness <= readFunction( i,
32          j))
33          loss++;
34        }
35      }
36    }
37    if ( loss <= maxLoss)
38      counter++;
39  }
40  return counter / maxWindow;
41 }
```

Listing 4.10: Function used to compute fuzzy values for ridges and peaks

In [Listing 4.10](#) it is possible to find the function used to compute the fuzzy value of a cell. The necessary parameters are:

multiHgt Javascript Object containing 9 HGT files (the one under anal-

ysis and the surrounding 8)

size Number of rows and columns of the HGT files

row Row of the cell to be analyzed

col Column of the cell to be analyzed

maxWindow Value that indicate how many scales are analyzed to compute the fuzzy value

steepness Value used to recognize only prominent ridges or peaks. It is the minimum elevation gap that must subsist between the elevation of the central cell and the others.

withLoss Boolean value that set to **true** makes the function perform the computation for peaks.

The computation for ridges proceeds in this way: first of all the elevation of the cell under analysis is read from the DEM and stored (line 12); immediately afterwards a counter is set to 0 in order to be ready to accumulate the number of times the analyzed cell is recognized to belong to a ridge. Then, through a for cycle that is repeated exactly **maxWindow** times, the cell is analyzed at different scales. To do that, at line 15, it is computed the maximum number of cells that, at the given scale, may not respect the condition on the elevation. A variable to count these cells is initialized too. Using four loops (lines 17 to 28) the condition that the elevation of the analyzed cell must be at least **steepness** meters higher than the others is checked against the cells forming the concentric square corresponding to the current scale (see [Figure 3.16](#)) As soon as the condition is violated more times than what established in **maxLoss** at line 15, the cycles are broken, otherwise the external counter is increased saying that, at the current scale, the cell possibly is part of a ridge. Once the cell is analyzed for all the scales, the average number of times that it is recognized to be a ridge is returned as the result.

The computation for the peaks is the same. The effect of setting **withLoss** to **false** is that **maxLoss** is 0; this means that **all** cells must be lower than the cell in analysis of at least **steepness** meters

Lines from 2 to 10, that remained unanalyzed, represent only an optimization for all the cells that are not on the border of the HGT file. In particular where it is certain that during the analysis of that cell are involved only cells of the same HGT file, a faster function is used to read the HGT file.

4.3.3 Grouping

Having only scattered fuzzy values is not enough to conduct the analysis but there is the need to aggregate them into areas; in order to do that it was decided to create groups of neighbors cells.

Grouping is performed checking values greater than a given threshold on matrices of fuzzy values and looking for neighbors. The most important functions are reported in [Listing 4.11](#).

```
1 function findGroupMembers( fullMatrix, groupsMatrix, row,
   col, groupId, fuzzyThreshold) {
2   for ( var i = row - 1; i <= row + 1; i++) {
3     for ( var j = col - 1; j <= col + 1; j++) {
4       if ((i != row || j != col) && fullMatrix.get([ i,
   j]) > fuzzyThreshold) {
5         fullMatrix.set([ i, j], 0);
6         groupsMatrix.set([ i, j], groupId);
7         findGroupMembers( fullMatrix, groupsMatrix, i, j,
   groupId, fuzzyThreshold);
8       }
9     }
10  }
11 }
12
13 function createGroups( analyzedMatrix, fullMatrix,
   groupsMatrix, size, fuzzyThreshold) {
14   analyzedMatrix.forEach( function( value, indexes) {
15     let row = indexes[0] + size;
16     let col = indexes[1] + size;
17     if ( fullMatrix.get([ row, col]) > fuzzyThreshold) {
18       fullMatrix.set([ row, col], 0);
19       groupId++;
20       groupsMatrix.set([ row, col], groupId);
21       findGroupMembers( fullMatrix, groupsMatrix, row, col,
   groupId, fuzzyThreshold);
22     }
23   }, true)
24 }
```

Listing 4.11: Functions used to create groups of cells

This procedure starts with **createGroups** function that takes the following parameters:

analyzedMatrix Matrix under analysis

fullMatrix Matrix composed by 9 matrices with the analyzed one at

the center.

groupsMatrix Matrix used to store groups IDs

size Number of rows and columns of matrices

fuzzyThreshold Minimum fuzzy value to be considered

Here there is a loop on non-zero values contained in the **analyzedMatrix** of fuzzy values. For each of these values is performed a check also in the **fullMatrix** adjusting the indexes; this is done because, in the **fullMatrix**, all the cells that are already assigned in a group are zeroed. In this way one cell can belong to only one group. If this condition is satisfied a new group ID is generated and assigned to that cell.

Now a recursive function (**findGroupMembers**) is used to find all the other members of the just created group. This function check if there is any neighbor and, if present, it is added to the group and launched again with the current cell as parameter in order to find, in turn, its neighbors.

4.3.4 Data Storage

In [subsection 4.3.1](#) has been already discussed how the Digital Elevation Model is stored into HGT files and the decision to derive the division in patches of 1 degree x 1 degree for the other files that the algorithm produces, but nothing has been explained about the files format.

Since the beginning of this section the data coming from the Digital Elevation Model or produced by the algorithm was considered as they were like grids or matrices because it is most natural way to think about them. With respect to the Digital Elevation Model, that is a grid full of values, the computed fuzzy values, and consequently groups IDs, are often zero and have a positive value only in places where the highest parts of the mountains lie. This situation is perfectly suitable for sparse matrices.

The implementation of sparse matrices that has been used is the one included in the *math.js* package (<http://mathjs.org/>). This allows to easily create sparse matrices, store them as a JSON files and load them again later.

Chapter 5

Evaluation

In this section, we present the preliminary results of the evaluation of the main contributions of the thesis, namely:

- the acquisition, processing and visualization of Points of Interest (specifically, huts) and
- the discovery of peaks from the data of the Digital Elevation Model of the Earth.

5.1 Alpine huts integration in PeakLens

The evaluation of acquisition, processing and visualization of Points of Interest (specifically, huts) has been conducted both qualitatively and quantitatively:

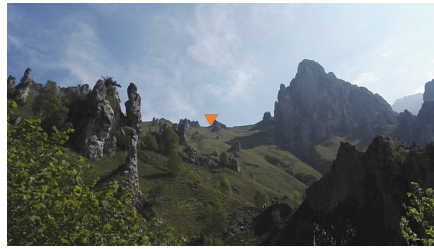
- Quantitative evaluation has followed a structured process aimed at estimating the accuracy of the positioning of the POI icons in the user interface. It is presented in details in [subsection 5.1.1](#).
- Qualitative evaluation has focused on the assessment of the quality of the user experience by a panel of prospect users and experts (the latter belonging to a professional GUI design firm). It is explained in detail in [subsection 5.1.2](#).

5.1.1 Quantitative Evaluation

In order to quantitatively evaluate the accuracy of the positioning of POIs in the mobile application, a process consisting of the following steps was followed:

- A test user records a sequence with PeakLens in real outdoor conditions, pointing at a landscape where one or more POIs should be visible. A sequence consists of a series of camera frames (typically 250), automatically annotated with the 2D screen positions of the objects (peaks and POIs) that appear in the camera view. Sequences can be recorded with the debug-oriented version of PeakLens.
- The sequence is uploaded in the Sequence Annotator online application, an existing web tool, developed for peak position assessment, whereby an evaluator can revise the sequence frame by frame and change manually the position of objects, so to shift them to the correct one. In this way, it is possible to produce a *gold sequence* from a recorded sequence, which can serve as the ground truth for the quantitative evaluation.
- The POI positions in the recorded sequence are contrasted with the corresponding ones in the gold sequence, in order to compute quantitative error metrics.

The aforementioned process was executed on 5 sequences represented by the images present in [Figure 5.1](#) in which the annotated huts are pointed by orange triangles.



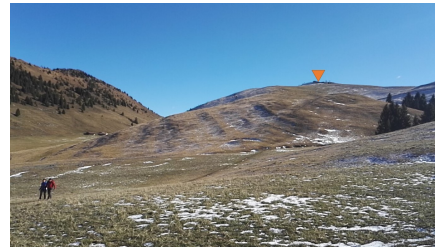
(a) Sequence 1: Rifugio Rosalba



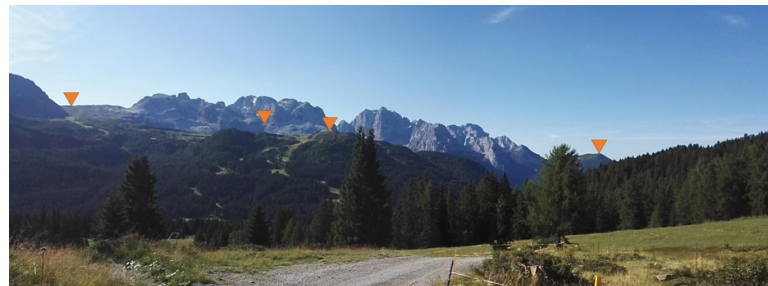
(b) Sequence 2: Bivacco Due Mani



(c) Sequence 3: Rifugio Menaggio, Rifugio La Preda, Rifugio La Canua



(d) Sequence 4: Rifugio Parafulmine



(e) Sequence 5: Rifugio Stoppani, Rifugio Tucket, Rifugio Spinale, Rifugio Doss del Sabion

Figure 5.1: Examples of ground truth images, extracted by an evaluator from the sequences recoded in the field

The results of the comparison between recorded and gold sequences are presented in [Table 5.1](#). For each sequence the Average Angular Error (AAE) is computed both for the case in which is used the sensor alignment and the global alignment. The AAE, considers the positioning errors of all the huts w.r.t. the position in the gold sequence [20].

Sequence Number	Number of Frames	Average Distance from Huts	Average Angular Error	
			Sensor Alignment	Global Alignment
1	94	541 m	7,69	3,05
2	207	1246 m	2,34	3,24
3	253	7398 m	3,41	1,25
4	329	601 m	5,29	7,24
5	298	6893 m	12,53	1,56
Average			6,25	3,3

Table 5.1: Average Degree Error

From the results in [Table 5.1](#) it is noticeable that the angular error denotes a good precision when the distance from the POIs grows, but, due to the distortion in the generation of the virtual panorama from the DEMs, when the distance is low the angular error increases. This problem is also reflected by the fact that the Global Alignment, in close distance sequence 2 and 4, worsen the results.

The assessment must be considered preliminary, because the following aspects have not been evaluated yet:

- Recall: the evaluation has not considered the the percentage of the acquired POIs with respect to the really existing POIs. Such an assessment would require as a gold standard an independent source with the complete database of the existing POIs, which is not publicly available in digital format. Indeed, one of the goals and future works of the thesis will be to solicit the users to insert their own POIs, completing the POI database until it gets as close to reality as possible.
- Local POI position adjustment: presently the positioning of POI depends only on: 1) the global alignment between the virtual panorama extracted from he DEM and the real skyline extracted from the image. 2) the 3D coordinates of the POI computed from the DEM and OSM data. Actually, no local adjustment of the position of the POI is done by analyzing the vicinity of the 2D scree coordinates in the frame. This implies that any error in the skyline alignment immediately impacts the positioning of the POI.

5.1.2 Qualitative Evaluation

Concerning the qualitative evaluation, the assessment has concentrated on the following quality factors:

- Graphic design of the POIs and of the POI labels.
- Appearance and placement of the POI labels, especially with respect to the interference with the labels of other objects such as peaks.
- Overall readability and clarity of the interface.

These concerns have been embedded into a questionnaire, deployed online and filled by PeakLens users. The questionnaire start with a demonstrative video that shows several sequences with huts.

The questionnaire contains the following statements:

1. The labels for huts are placed correctly on the relative elements
2. The labels of the huts and the peaks are distinguishable
3. It is useful to distinguish the huts' icon from the peaks' one
4. The transparency of the labels helps the visualization of elements
5. The labels of peaks and huts overlap
6. The names of peaks and huts are clearly readable
7. The new labeled elements does not make the visualization of peaks less clear
8. Adding these new elements makes the App more useful

At the end of the questionnaire, a free text question permitted the insertion of suggestions on how to improve the visualization.

The responses were graded using the Likert Scale, which typically proposes these responses [27]:

1. Strongly disagree

2. Disagree
3. Neither agree nor disagree
4. Agree
5. Strongly agree

Figure 5.2 presents the results of the assessment questionnaire filled in by 10 people.

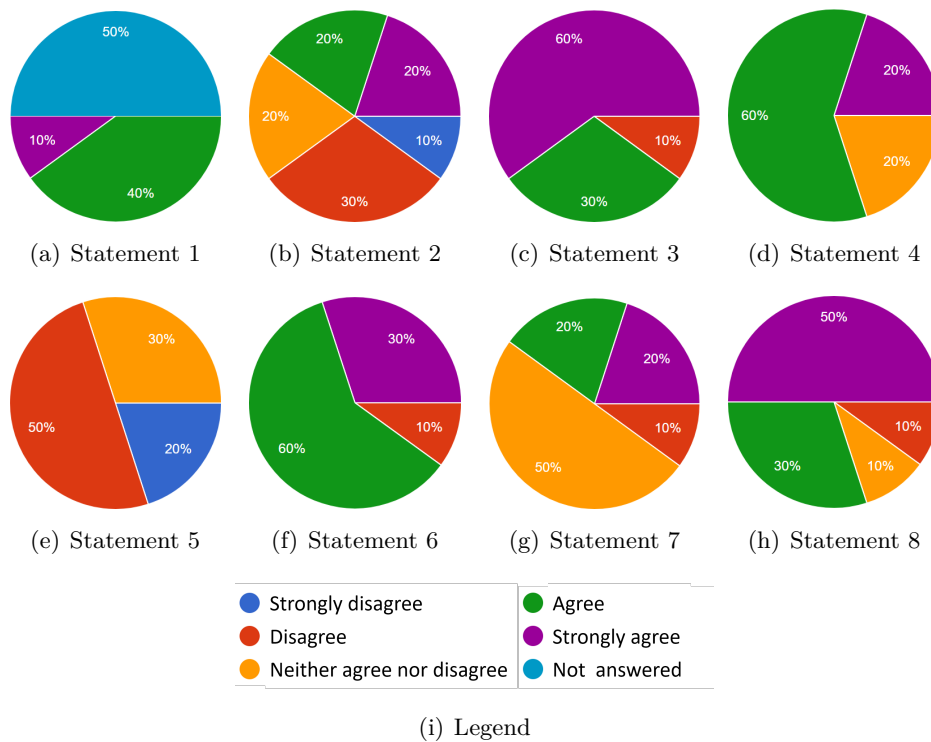


Figure 5.2: Assessment Questionnaire Results

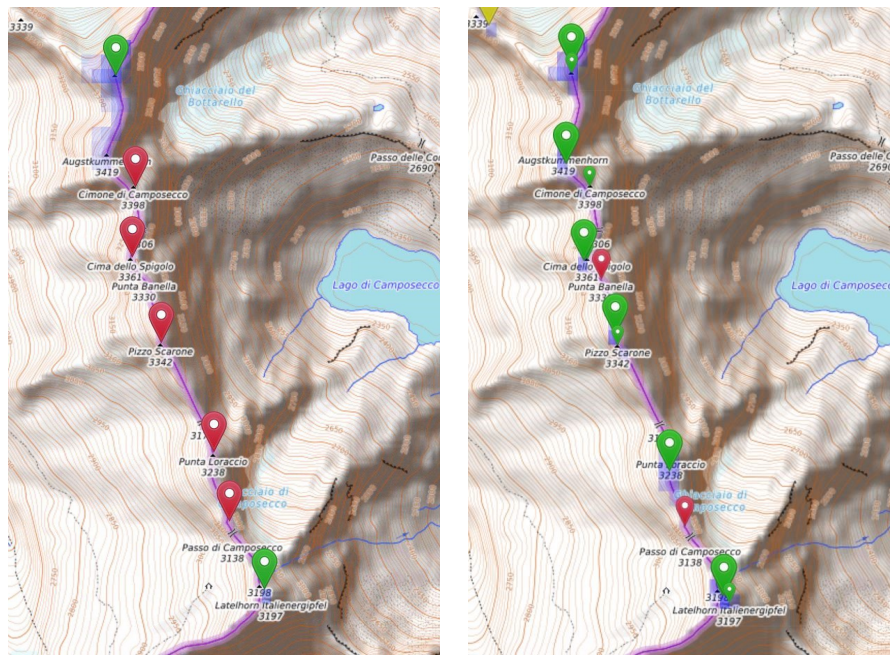
In general, the evaluators appreciated the new feature of identifying and visualizing huts; more work has to be done regarding the distinction between labels: among the suggestions, it was found that it may be useful to use different colors and not only different icons. The results also emphasize the fact that labels are readable, but it could be useful to improve the pagination not to worsen peaks visualization, even though it emerged that the labels of the two types of objects rarely overlapped.

5.2 Peaks discovery

This section presents the results of the improved peak extraction algorithm developed in the thesis, compared to the baseline from the literature: visual representations of the extraction and statistics to determine the quality of the outcome are shown.

5.2.1 Improvements achieved over the baseline method

This section presents, by means of juxtaposed images, the improvements achieved in the design of an optimized version of the procedure for detecting peaks from DEM data; it compares the results of the developed algorithm with respect to the baseline method explained in [23].



(a) Using the algorithm explained in [23] (b) Using the algorithm developed in this thesis

Figure 5.3: Improvements in peaks recognition on ridges

Figure 5.3 shows an example of peaks extraction in a ridge area: the baseline algorithm identifies only 2 out of 8 peaks; in contrast, our improved version

recognized 6 out of 8 peaks. This happens because, once the ridge area is identified, our algorithm performs a more specific analysis on smaller windows, thus finding also the peaks that otherwise would be too much in proximity of other peaks in the ridge and that, as a consequence, would not satisfy the conditions of prominence with respect to the adjoining DEM cells.



(a) Using the algorithm explained in [23] (b) Using the algorithm explained in this section

Figure 5.4: Improvements in peaks recognition in non-mountain areas

Figure 5.4 shows the application of the peak identification algorithms to the city of Dubai. Note that skyscrapers are incorrectly recognized as peaks in 5.4(a) but not in 5.4(b). This happens thanks to the preliminary filtering done by identifying ridge areas: a skyscraper likely appears within a small number of DEM3 cells and, because of this, it would not be considered as a ridge (given a reasonable S_r parameter) and the area would be discarded before extracting peaks.

5.2.2 Performance Evaluation

The peak discovery task is an information retrieval task, which queries a data store (the DEM) to extract relevant items (the peaks). Therefore, evaluation can follow the standard procedures for assessing the quality of a data retrieval task, based on the evaluation of precision and recall. Evaluating the aforementioned metrics requires a ground truth data set, which consists of a geographic region and all the peaks contained in it.

Our evaluation has exploited two different approaches for assessing the truth value of an extracted peak:

- We have used Switzerland as a region for testing and exploited the high quality peak data set SwissNames3D[13] as a proxy to the real peaks present in the region.
- To cope with regions where no high precision peak data set is available, we have built a crowdsourcing application that allows expert users to:
 - Confirm that a retrieved peak is a really existing peak (true positive).
 - Specify that a retrieved peak is not a really existing peak (false positive).

The SwissNames3D dataset was used to analyze how the results change, depending on the algorithm characteristic parameters:

Figure 5.5 represents how precision and recall are influenced by alternative configurations of the parameters (reported in Table 5.2). In general, an increment of parameters that boost the prominence of given cells (P_r , F_r , P_p and F_p) yields higher precision, i.e. only prominent cells are categorized as peaks; conversely, an increment of parameters that regulate the number of cells analyzed (W_r and W_p) corresponds to an higher recall, i.e., by analyzing a larger area the probability of finding "less prominent" cells than the currently analyzed one is higher.

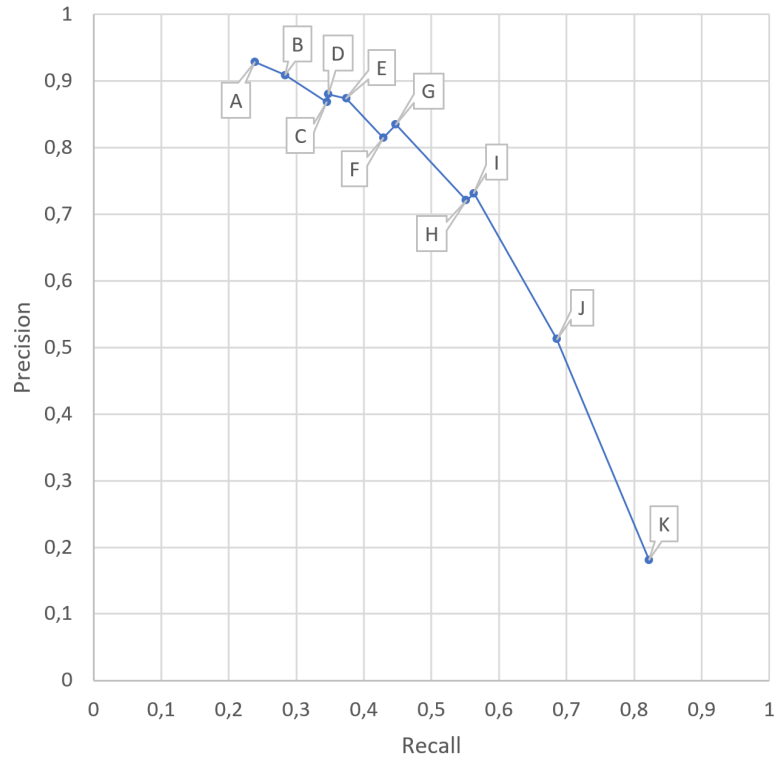


Figure 5.5: Precision-Recall Curve (configurations and values are shown in Table 5.2)

	P_r	W_r	F_r	S_r	P_p	W_p	F_p	Precision	Recall
A	50	5	0.2	5	50	5	0.5	92.88	23.87
B	50	6	0.2	5	50	6	0.5	90.88	28.37
C	40	5	0.15	5	40	5	0.4	86.85	34.53
D	40	6	0.15	5	40	6	0.4	88.04	34.75
E	40	7	0.15	5	40	7	0.4	87.34	37.4
F	30	6	0.125	5	30	6	0.3	81.48	42.9
G	30	7	0.125	5	30	7	0.3	83.44	44.72
H	20	7	0.1	5	20	7	0.2	72.13	55.13
I	20	8	0.1	5	20	8	0.2	73.13	56.3
J	10	5	0.1	5	10	10	0.2	51.27	68.56
K	0	8	0.05	5	0	8	0.1	18.07	82.25

Table 5.2: Values used for Figure 5.5 (See section 3.3 for their description)

Analyzing the results of the extraction in the Switzerland region, it can be noticed that a configuration that results in a high precision is useful to

find a restricted number of real peak candidates. On the other hand, a configuration characterized by lower precision and higher recall helps spot misplaced peaks (e.g., wrongly user-inserted peaks in OpenStreetMap); in fact, if the area where a peak was placed in OSM is not classified as a peak by the algorithm with very loose parameters, it may be an error.

As a consequence, the optimal parameter configuration depends on the context and objectives with which the algorithm is used.

Considering in more detail two different configurations from [Table 5.2](#):

- [Table 5.3](#) shows the results in case the parameters are set to the configuration A of [Table 5.2](#). The requirements for a feature to be considered a peak are high and this results in only a few false positives. At the same time, many peaks present in the dataset are not found creating many false negatives. Having a small number of false positives yields high precision, while, because of the high number of false negatives, the recall decreases.
- [Table 5.4](#) shows the results in case the parameters (presented in configuration J of [Table 5.2](#)) to consider a feature a peak are less stringent: recall is higher because many features are classified as peaks and a great part of them corresponds to the peaks from the source dataset; the consequence is that the precision decreases compared to the previous settings because of the number of false positives.

TRUE POSITIVES	1643
FALSE NEGATIVES	5240
FALSE POSITIVES	126
PRECISION	92.88
RECALL	23.87
F1 SCORE	37.98

Table 5.3: Extraction statistics using strict parameters (Configuration A of [Table 5.2](#))

TRUE POSITIVES	4719
FALSE NEGATIVES	2164
FALSE POSITIVES	4485
PRECISION	51.27
RECALL	68.56
F1 SCORE	58.66

Table 5.4: Extraction statistics using loose parameters (Configuration J of [Table 5.2](#))

The number of false positives in Switzerland area is pretty small, but in areas where the mapping is underdeveloped the number of false positives increases significantly (as shown in [Table 5.5](#)).

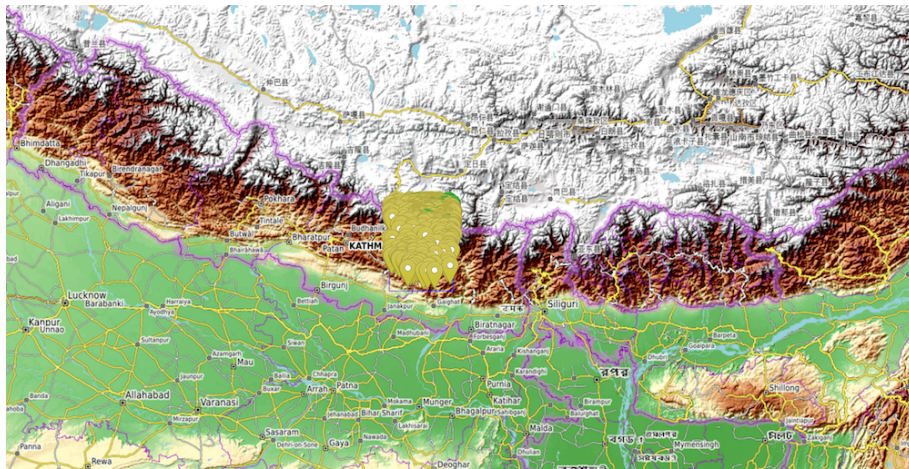


Figure 5.6: Area of extraction - Around Everest, Himalaya

TRUE POSITIVES	48
FALSE NEGATIVES	31
FALSE POSITIVES	1973
PRECISION	2.38
RECALL	60.76
F1 SCORE	4.58

Table 5.5: Extraction statistics using strict parameters for Everest area, Himalaya

Parameter	Value
P_r :	50
W_r :	5
F_r :	0.2
S_r :	5
P_p :	50
W_p :	5
F_p :	0.5

Table 5.6: Strict extraction parameters for Everest area, Himalaya (See [section 3.3](#) for their description)

Evaluating the results in the area shown in [Figure 5.6](#), it was possible to notice that a number of peaks classified as false positives were, in fact, real peaks, because they were present in other datasets. In many other cases, it was possible to extract peaks that were not on any dataset available on the Internet, but also after a direct analysis seemed to satisfy the conditions of being classified as a peak.

To cope with the deficiency of high quality publicly available peak data sets, an instrument to investigate the results was developed in order to allow mountain experts to see the results directly on a 3D rendering of the peak's area.

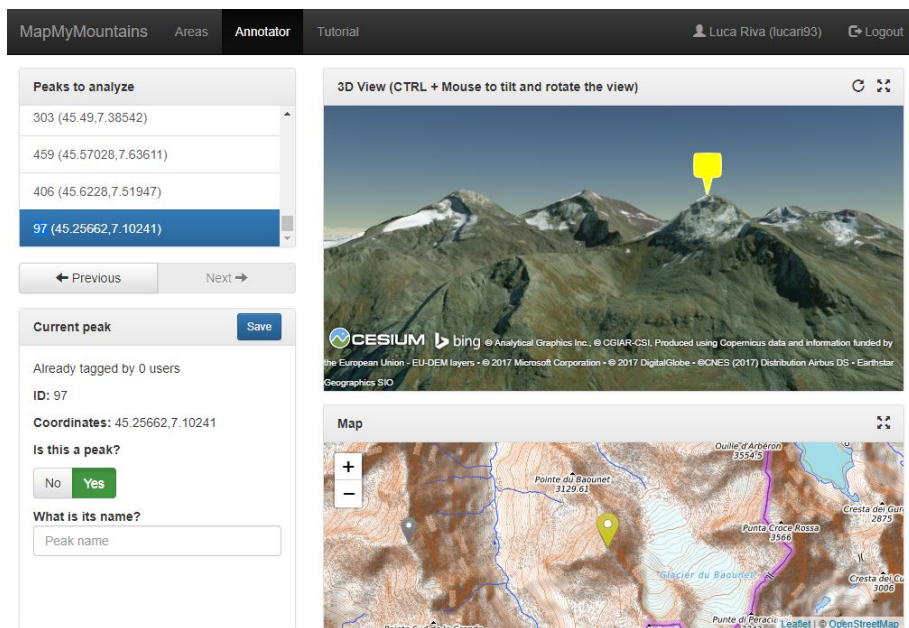


Figure 5.7: Expert Crowdsourcing Interface

The interface, presented in Figure 5.7, is mainly constituted by four windows:

- The top left window presents a list of the false positives found by the algorithm, which may be misclassified due to the absence of proper data in the publicly available peak data set. The user may select any of them to view the relative placeholder on a 2D and on a 3D map and confirm (or not) the fact that the result is really a peak.
- The bottom left window asks to answer simple questions about the current peak, the user may specify whether they think that the result is a peak or not and possibly specify its name.
- The top right window presents a 3D view of the current extracted peak area. The user may use the mouse to change the point of view (rotation and inclination).
- The bottom right windows presents a 2D view of the current extracted peak area, including also other peaks in the same area. The user may also click on one of the other peaks placeholders to select another peak to analyze.

To do that, an open source library called Cesium[1] was used.

All the answers given by the user are saved in a SQL database, which gives the possibility to analyze and summarize the results of the expert crowdsourcing campaign.

The target of this crowdsourcing tool are mountain experts; to reach such experts Facebook was used to publish a post asking for help to improve OSM data. The selected areas provided to these users are:

- Dolomites
- Alpi Cozie
- Valleys of Lanzo, Orco and Soana

This areas were proposed to the Facebook groups relative to such areas (see Table 5.7), asking for collaboration and illustrating how their contribution may improve the projects sustained by OSM.

Area	Group Name	Members
Dolomites	DoloMitici!	53439
	val badia che passione!	14840
	Dolomiti...una Passione	26700
Alpi Cozie	ALPI COZIE e dintorni ...dal 2009 - nuovo look -	8798
Valleys of Lanzo, Orco and Soana	Montagne delle Valli di Lanzo, Orco e Soana	12481

Table 5.7: Facebook Groups Statistics

The crowdsourcing effort is ongoing at the time of writing; we will report the status of the crowd contribution at the discussion of the thesis.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

The thesis has faced the problem of acquiring, processing and displaying geo-referenced data for use in augmented reality mobile applications. Specifically, the general problem has been instantiated on a concrete use case dealing with the collection, augmentation, validation and display of data about peaks and points of interest (huts) used by PeakLens, an outdoor mobile applications for mountain trekkers.

This work has led to the development of a set of modules based on the need to display new type of elements besides peaks in PeakLens and improve the general quality of data. The contributions achieved in the thesis can be summarized as follows:

- A new module able to download geographical data from OSM, enrich the elements with alternative names retrieved from WikiData and validate the elevation information using the Digital Elevation Model was created.
 - 120 thousands alternative names were added making the data convenient for international use
 - 100 thousands peaks, which have no specified elevation on OSM, were added to the visualization extracting their elevation directly

from the DEMs.

- 4 thousands errors of elevation where corrected avoiding to show bad information to the user
- The data produced by the module presented in the previous point is at the base of the PeakLens application that in this work was modified in order to display Alpine and Wilderness Huts together with Peaks in augmented reality on the screen of an Android Smartphone:
 - almost 13 thousands alpine huts were added
 - more than 6 thousands wilderness huts were added
 - a new method to avoid overlapping of different element's labels was studied

This was possible also thanks to algorithms already present able to generate a Virtual Panorama and align it with the camera image.

- A module was created in order to discover new Peaks from the analysis of the Digital Elevation Model. This one paved the way for the integration of new validating modules and crowdsourcing systems able to generate new data or correct the already present ones.

6.2 Future work

This thesis has scratched the surface of the many ways in which the PeakLens data collection, processing and visualization could be improved. Future work could pursue the following directions:

- Import and manage non punctual elements, e.g. lakes (polygon) and trails (polyline)
- Improve the quality of the peaks extraction from DEM using a combination of algorithms
- Use crowdsourcing to enrich the data by creating a simple interface that allows the users to contribute adding names and confirming the peaks extraction results
- Help the OSM community by adding the data corrected and added by the use of the algorithms explained in the previous sections

6.2.1 Managing non punctual elements

Presently, PeakLens manages and visualizes peaks, which are point type elements identified by only one coordinate, and huts, which are characterized by more than a point, but can be assimilated to punctual objects, because it is natural to identify their midpoint in order to place their label on the screen.

However, in PeakLens there is the technical possibility of acquiring and saving elements that are not identified by only one coordinate: this offers a chance for expanding the AR visualization to non punctual objects.

In the future, it may be interesting to introduce other elements, like towns and lakes, which occupy a large part of the user vision and, as a consequence, would need a more advanced algorithm to place the label: elements such as these may be visible even if a portion of them is hidden behind other elements (e.g. a lake which is partially covered by an occlusion or mountain slope); also in this case, PeakLens should be able to detect this situation and place the label where the user has a clear view of the element concerned. To do so, one of the possibilities could be to calculate a series of relevant points of the element and choose one which one is not occluded by another element.



Figure 6.1: Possible approach to non punctual elements label positioning: characteristic points distributed along the element are saved in the PeakLens server, one among the ones that are not occluded is chosen to be labeled

There are also other types of elements to be introduced like footpaths or rivers. These types of elements are represented by polylines that may be drawn entirely where they are visible and then place a label near one of their points.

As it happened for the huts, the points relevant for a label to be placed

or a line to be drawn should be aligned relatively to the peaks, in fact it is important to remember that PeakLens strength depends greatly on the alignment made not only by sensors, but also through the matching between the virtual panorama and the one seen by the smart-phone camera.

The addition of other elements to the application will make it necessary to add a POI filter in order to select what to be shown in augmented reality and not overcrowd the screen.

6.2.2 Applying other algorithms to extract peaks from DEM

Many algorithms used in order to extract features from DEM exist, in this work the problem was faced using a variant of the algorithm proposed in [23]. Other algorithms of machine learning were presented, like the one in [25], which apply deep learning and convolutional neural networks to the problem.

As stated in [subsection 2.2.3](#) using more than one method to discover morphological objects and integrating the results may improve the precision already obtained. In the future, it is worth trying to understand the contribution of these new techniques.

6.2.3 Discover new peak names through crowdsourcing

As stated in [chapter 3](#) one of the purpose of this work is to automatically extract peaks analyzing the Digital Elevation Model. The algorithm implemented to reach this objective is actually part of an independent module than can be used for multiple purposes.

The crowdsourcing interface, developed for the experimental study explained in [section 5.2](#), is only the starting point. The ability to create a 3D reconstruction of the area surrounding the candidate peak gives the possibility to create different type of applications to receive contribution from users. An example could be a social media application that presenting the 3D view of a discovered peak, asks the user whether it is actually a peak or not and eventually also the name or a correction of its position.

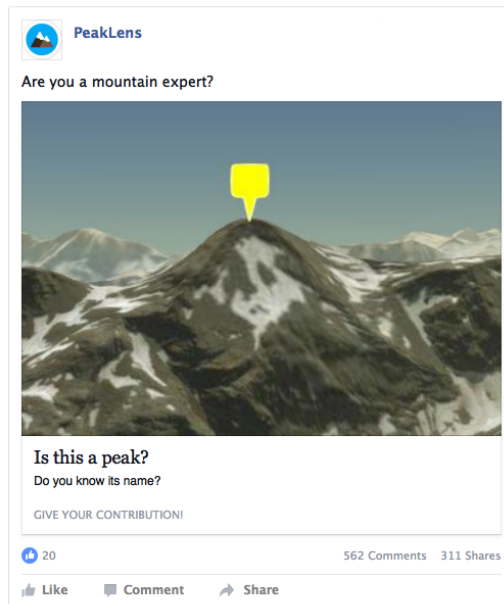


Figure 6.2: Social Media Crowdsourcing Mockup

In the future it would be relevant to include directly in PeakLens the new found peaks presenting them to the user in augmented reality without a name. The user, at this point, should be able to click on the peak with a missing name and eventually add it. At the same time, given the fact that the feature extraction algorithm will always have room for errors, the user should also be able to signal a false positive to PeakLens, which will have to deal with this information, for example deleting the found peak if there is more than one user reporting.

6.2.4 Improving data directly on OSM

During this work a special consideration to improve the quality of the data was always present. This led to include an elevation check algorithm, the integration of alternative names from other sources and a development of a method to discover new peaks. Moreover, this work allows to implement crowdsourcing systems able to increment the quality of the data. The amount of data corrected and produced from all these procedures can be very high over time.

At the time of writing the data improved by PeakLens are stored for internal

use only. One of the improvements planned for the future is to automatically update OpenStreetMap with the corrected and improved information about already present elements as well as update it with new ones.

OSM change-set are obviously relative to a user, as a consequence PeakLens will have to mark its updates identifying itself as a bot and the precision of such updates should be high in order to build a proper reputation.

Bibliography

- [1] Cesium. <https://cesiumjs.org/>. Accessed: 2017-10-11.
- [2] Eventstream - a toolkit to make creating and working with streams easy. <https://github.com/dominictarr/JSONStream>. Accessed: 2017-11-04.
- [3] Jsonstream - streaming json.parse and stringify. <https://github.com/dominictarr/JSONStream>. Accessed: 2017-11-04.
- [4] Nasa srtm frequently asked questions. <https://www2.jpl.nasa.gov/srtm/>. Accessed: 2017-11-20.
- [5] Nasa srtm frequently asked questions. <https://www2.jpl.nasa.gov/srtm/faq.html>. Accessed: 2017-10-18.
- [6] Open street map. https://wiki.openstreetmap.org/wiki/About_OpenStreetMap. Accessed: 2017-10-10.
- [7] Open street map elements. <http://wiki.openstreetmap.org/wiki/Elements>. Accessed: 2017-10-30.
- [8] Overpass API for OSM read-only server for osm. http://wiki.openstreetmap.org/wiki/Overpass_API. Accessed: 2017-10-10.
- [9] Peak finder app. <https://www.peakfinder.org/mobile/>. Accessed: 2017-10-11.
- [10] Peaklens - mountain identification android mobile app. <http://peaklens.com/>. Accessed: 2017-08-14.
- [11] Rfc7946 - the geojson format. <https://tools.ietf.org/html/rfc7946>. Accessed: 2017-11-7.
- [12] Snowwatch portal. <http://snowwatch.polimi.it/>. Accessed: 2017-08-14.

- [13] swissnames3d - swissnames3d is the most comprehensive collection of swiss geographical names. <https://shop.swisstopo.admin.ch/en/products/landscape/names3D>. Accessed: 2017-11-27.
- [14] View ranger frequently asked questions. <http://support.viewranger.com/index.php?pg=kb.page&id=200>. Accessed: 2017-10-11.
- [15] View ranger home page. <http://www.viewranger.com/en-US>. Accessed: 2017-10-11.
- [16] Viewfinder panoramas. <http://viewfinderpanoramas.org/>. Accessed: 2017-11-13.
- [17] Wikidata introduction the free knowledge base with 37,406,242 data items that anyone can edit. <https://www.wikidata.org/wiki/Wikidata:Introduction>. Accessed: 2017-10-11.
- [18] Wikidata main page. https://www.wikidata.org/wiki/Wikidata:Main_Page. Accessed: 2017-10-11.
- [19] Ronald T Azuma. A survey of augmented reality. *Presence: Teleoperators and virtual environments*, 6(4):355–385, 1997.
- [20] Carlo Bernaschina, Roman Fedorov, Darian Frajberg, and Piero Fraternali. A framework for regression testing of outdoor mobile applications. In *Proceedings of the 4th International Conference on Mobile Software Engineering and Systems*, pages 179–181. IEEE Press, 2017.
- [21] Mark Billinghurst, Adrian Clark, Gun Lee, et al. A survey of augmented reality. *Foundations and Trends® in Human–Computer Interaction*, 8(2-3):73–272, 2015.
- [22] Roman Fedorov, Darian Frajberg, and Piero Fraternali. A framework for outdoor mobile augmented reality and its application to mountain peak detection. In *International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pages 281–301. Springer, 2016.
- [23] Peter Fisher, Jo Wood, and Tao Cheng. Where is helvellyn? fuzziness of multi-scale landscape morphometry. *Transactions of the Institute of British Geographers*, 29(1):106–128, 2004.
- [24] Darian Frajberg, Piero Fraternali, and Rocio Nahime Torres. Convolutional neural network for pixel-wise skyline detection. In *International Conference on Artificial Neural Networks*, pages 12–20. Springer, 2017.

- [25] Xiangyun Hu and Yi Yuan. Deep-learning-based classification for dtm extraction from als point cloud. *Remote Sensing*, 8(9):730, 2016.
- [26] Antonio La Salandra. Immersive mobile augmented reality application for mountain peak recognition, 2017.
- [27] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [28] RJ Pike, IS Evans, and T Hengl. Geomorphometry: a brief guide. *Developments in Soil Science*, 33:3–30, 2009.
- [29] Boleslo E Romero and Keith C Clarke. Exploring uncertainties in terrain feature extraction across multi-scale, multi-feature, and multi-method approaches for variable terrain. *Cartography and Geographic Information Science*, pages 1–19, 2017.
- [30] JOSEPH Wood. Scale-based characterisation of digital elevation models. *Innovations in GIS*, 3:163–175, 1996.