POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO
Master of Science in Computer Engineering

Development of an Android Mobile Application for Mountain Peak Detection

Supervisor: Prof. Piero Fraternali
Assistant Supervisor: Prof. Marco Tagliasacchi

Master Graduation Thesis by:
Simone D'Agnano
id. 801120

Academic Year 2014/15

POLITECNICO DI MILANO
Scuola di Ingegneria dell'Informazione

POLO TERRITORIALE DI COMO
Corso di Laura Specialistica in Ingegneria Informatica

Development of an Android Mobile Application for Mountain Peak Detection

Relatore: Prof. Piero Fraternali
Correlatore: Prof. Marco Tagliasacchi

Tesi di laurea di:
Simone D'Agnano
matr. 801120

Anno Accademico 2014/15

Abstract

We present an Android application able to detect the peaks of the surrounding mountains visible from the user point view in that exact position. The system exploits the on board sensors as the GPS, the magnetometer, the accelerometer and the gyroscope to find the geographical location, the cardinal directions and the axis orientation of the device. Through the camera the images of the visible mountains are acquired and showed in the camera preview. The detection of the correct positions of the peaks passes through several phases. First we harness the sensors to obtain the related synthetic panorama recovered from the Digital Elevation Model (DEM) related to that current geographical area. After the peak names are identified and visualized in the camera preview exploiting the axis orientation and the information received together with the panorama. At the end this stage a deeper content analysis on the images captured from the camera is performed exploiting algorithms of digital image processing about object recognition to improve the precision reached from the sensors. The final experience is proposed in Augmented Reality Environment allowing an interactive real-time visualization with high precision level. In this work several deriving applications are discussed as the monitoring of environmental resources and the construction of collective intelligence system designated to the construction of environmental models and massive social media collections.

Sommario

Si presenta una applicazione Android in grado di riconoscere i picchi delle montagne circostanti visibili dall'utente dalla specifica posizione in cui si trova. Il sistema sfrutta i sensori a bordo del dispositivo come il GPS, il magnetometro, l'accelerometro e il giroscopio per poter ricavare la posizione, le direzioni cardinali e l'orientamento degli assi del telefono. Attraverso la camera vengono acquisite le immagini delle montagne visibili dall'utente poi direttamente visualizzate nell'anteprima della camera. L'individuazione della corretta posizione dei picchi all'interno delle immagini passa attraverso varie fasi. Inizialmente vengono sfruttati solamente i sensori per ottenere il corrispettivo panorama sintetico ricavato dal Modello Digitale di Elevazione (DEM) relativo all'area geografica corrente. Successivamente vengono identificati i nomi delle vette nell'anteprima attraverso etichette sfruttando l'orientamento degli assi e le informazioni ricavate dalla rete insieme al panorama. Terminata questa fase viene effettuata un'analisi approfondita del contenuto delle immagini catturate dalla camera attraverso algoritmi di processamento di immagini per poter migliorare la precisione basata su sensori. L'esperienza finale proposta in Realt Aumentata consentendo una visualizzazione interattiva real-time e ad alta precisione. Nell'elaborato sono discusse le diverse applicazioni che possono derivarne come il monitoraggio delle risorse ambientali e la costruzione di sistemi di intelligenza colletiva volte ad esempio alla costruzione di modelli ambientali e raccolta di ingenti collezioni multimediali.

Acknowledgements

Foremost, I would like to thank Prof. Piero Fraternali and Prof. Marco Tagliasacchi for the opportunity to work with them on a such interesting theme and fascinating project, and the support during the preparation of this thesis.
I would like to thank also:

- Dr. Roman Fedorov for helping with the preparation of this document.
- Dr. Ulrich Deuschle (http://www.udeuschle.de) for his kind permission of using his mountain panorama generating web tool.
- All my friends and colleagues who have helped and contributed to this work.

Lastly, but most importantly, I would like to thank my parents, Francesco and Elisabetta, my brothers, Valentina and Luca, and Chiara for constant support and motivation.

Contents

1 Introduction **1**
- 1.1 Human computation . . . 1
- 1.2 Augmented Reality . . . 3
- 1.3 The MountainWatch Project . . . 4
  - 1.3.1 Environmental Monitoring System . . . 5
  - 1.3.2 Innovation . . . 6
  - 1.3.3 Potential Impact . . . 7
- 1.4 Document Structure . . . 8

2 Related Work and State of the Art **9**
- 2.1 Mountain Peak Detection . . . 9
  - 2.1.1 Sensor based Mobile Apps . . . 9
  - 2.1.2 Offline Desktop based System . . . 10
- 2.2 Collective Intelligence . . . 12
  - 2.2.1 Game with a purpose . . . 12
- 2.3 Augmented Reality . . . 13
  - 2.3.1 Augmented Reality Systems . . . 13
  - 2.3.2 Augmented Reality On Mobile Devices . . . 14

3 Problema Statement and Proposed Approach **17**
- 3.1 The Motivations . . . 17
- 3.2 The Proposal . . . 18
- 3.3 The Application . . . 18
  - 3.3.1 Services initialization . . . 19
  - 3.3.2 GPS Localization . . . 21
  - 3.3.3 Building the Mountain Model and Sensor Matching . . . 22
  - 3.3.4 Global Matching . . . 27
  - 3.3.5 Local Matching . . . 30
  - 3.3.6 Tracking . . . 31
- 3.4 Possible Areas of Application . . . 32

4 Implementation Details **35**
4.1 Adopted Sensors and Functioning 35
4.2 OpenCv LIbraries 40
4.3 The Activities 40
4.4 Sensor Managers 41
4.5 Camera Handlers 42
4.6 The MountainOverlay View 43
4.7 The Mountain Downloader 44
4.8 Location Updaters 46
4.8.1 Global Location Updater 47
4.8.2 Local LocationUpdater 50
4.8.3 FollowPatch LocationUpdater 51

5 Experimental Study **53**
5.1 Data sets 53
5.1.1 Collecting Samples 54
5.1.2 Manual Alignment 55
5.2 Metrics 56
5.2.1 Test Flow Execution 57
5.2.2 Operating parameters 58
5.3 Results 58

6 Conclusions and Future Work **65**
6.1 Future Enhancements 65
6.1.1 Matching Performances 66
6.1.2 Local Digital Elevation Model 66
6.1.3 Safeguard and Environmental Modeling 67

Bibliography **69**

List of Figures

1.1 Proposed taxonomy of human computation including all related paradigms. 2

3.1 Application Schema of the phases composing the entire process. The color choice helps to understand the evolution of process: the first rough sensor peak estimate (white), the alignment based on first global content analysis (blue) and on the second local content analysis (light blue) and the ending phase of tracking based on the following patch matching (green). 19
3.2 Compass Calibration: how to move the device to help the sensor calibration at the application boot. 20
3.3 Extracted Patch from Panorama 2D with mountain markers. 21
3.4 The peaks are received from the network are in pixel coordinates in the Panorama 2D Frame. To display the peaks in the camera preview three transformations are applied passing from Panorama 3D Frame, Camera 3D Frame and Camera 2D Frame. 22
3.5 The overlapping of Panorama 2D and Camera 2D adapted from [1] 23
3.6 Cylindrical Coordinate System applied to rotational axis and geographical directions of testing mobile device. 23
3.7 The Original Panorama 3D Frame on the left and Calculation System on the right. 24
3.8 Compass Matching Result: the white marker represents the peak estimate only based on embedded sensors. The green marker is the estimate obtained from the groundtruth. 27
3.9 Sobel filters adopted. 28
3.10 Global Matching Result: the blue marker represents the peak estimate after the Global Matching Algorithm. The green marker is the estimate obtained from the groundtruth. 29
3.11 Local Matching Result: the light blue marker represents the peak estimate after the Local Matching Algorithm. The green marker is the estimate obtained from the groundtruth. 30

3.12 A patch retrieved in two different frames during the same sequence. 32

4.1 OnePlus One, the device adopted to test and analyze the application. 36
4.2 Smartphone rotational axis: the rotational angles among the x,y and z axis are respectively azimuth, pitch and roll. 38
4.3 Accelerometer and Gyroscope functioning 38
4.4 Coriolis Effect force in play 39
4.5 Sequence Diagram representing the execution flow of the Activities. 40
4.6 Class Diagrams of LocationManager, OrientationManager and NetworkManager. 41
4.7 Class Diagrams of CameraPreview and CameraShutter. . . . 42
4.8 Class Diagrams of MountainOverlay. 43
4.9 Class Diagrams of MountainDownloader. 44
4.10 The Graph represents the alignment background processes based on LocationUpdaters. 46

5.1 Samples taken from the Dataset. 54
5.2 The alignment of the frame with the panorama. 55
5.3 The frame is stretched to overlap perfectly the panorama. . . 56
5.4 The Confusion Matrix allows visualization of the performance of the algorithm: P and R are respectively the Precision and the Recall. 57
5.5 The Confusion Matrix representing the entire Dataset. 61
5.6 The classification of all the resulting peaks divided in four categories. 61
5.7 The Bar Chart represents the resulting DMD Error of the entire Data Set. 62
5.8 The Bar Chart represents the resulting DMD Error of the entire Data Set. 63

List of Tables

4.1 OnePlus One Accelerometer Specifications. 36
4.2 OnePlus One Gyroscope Specifications 37
4.3 OnePlus One Magnetometer Specifications 37

5.1 Degree Error Classification. 58
5.2 Operating parameters (defaults in bold). 58
5.3 Invariant operating parameters 59
5.4 Result obtained in the Sequences after the Sensor Matching phase. 59
5.5 Result obtained in the Sequences after the VCC Matching phases. 60

Chapter 1

Introduction

The main related topics suitable for this work are crowdsourcing, social computing and collective intelligence. These paradigms are correlated one to each other under certain aspects and they differ from others. Given this ambiguity due to the newness of these computer science themes in this chapter definitions of these basic concepts are explained, stressing the aspects covered directly by this work. The application purpose is defined presenting the innovative idea of this project. At the end the structure of the entire document is outlined.

1.1 Human computation

Human computation can be defined as:

> *" a paradigm for utilizing human processing power to solve problems that computer cannot yet solve. "* Alan Turing [2]

The human effort can be harnessed by means of enjoyable tasks with very simple access as from the vast internet population. Often the required operations exploit the human abilities that for computer are really tough. The tasks are often proposed presenting a multi-step processes involving human collaboration and building systems exposing operations as simple casual games. As presented in [3] the concept of human computation can be resumed with two main properties that must be satisfied to consider a system a human computation based system. First to be a human computation the problems faced from people have to be a computational tasks with the possibility that in future can be solvable from computers. Secondly the human participation is managed by the information system.

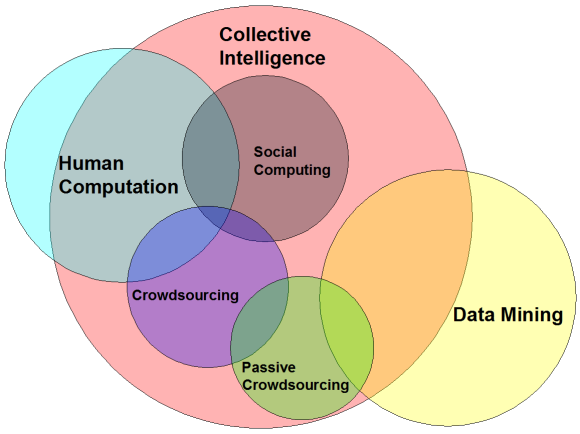

Figure 1.1: Proposed taxonomy of human computation including all related paradigms.

This particular work is nestled in a more general system which aims to build an environmental control system in a innovative way harnessing the spreading technologies blossomed in the last recent years, the smartphones. Thanks to our mobile application a new immediate acquisition channel becomes possible to populate the data set without the need of fixed and costing equipment already existing but poorly spread. A cheaper but capillary amount of data is available exploiting the user sharings and uploads. With respect to the two main properties defined before our system performs an hard calculation effort to detect and to identify the peaks inside the photographs satisfying the first property. About the human participation the system doesn't managed it directly since the users autonomously decide to takes shutters and then to share them. There are several fields that fall in the human computation. A possible taxonomy, displayed in the Figure1.1, is presented covering all the related researching field of our application combining definition given by [4] and by [3]:

- *Crowdsourcing*: This concept, deriving from outsourcing, is the act of taking a job traditionally performed by a designated agent (usually an employee) and outsourcing it to an undefined, generally large group of people in the form of an open call [3]. Our system is based on this concept since we involve people to use our application and indirectly we exploit their effort, the shutters, for further environmental analysis.

- *Social Computing*: In system based on social computing people are facilitated to share and to exchange multimedia information by means of information technology systems. The scopes can be broad and various but in general they do not involve computation but social interactions. In particular our system is a social computing example helping people not only to identify peaks but also stimulating the social interaction.

Examples can be share multimedia information, as the photographs, and give a contribute to the environment safeguard.

- *Data Mining*: Systems based on this paradigm have in general the purpose to extract interesting information from a large amount of data through the application of specific ad hoc algorithms. In our context all the entire gathered collection do matter and we do not need to distinguish valuable information from large collection. We analyze the whole resulting photographs already processed so our application does not turn out to be a typical data mining system.

- *Collective Intelligence*: Collective intelligence are platforms where groups of people collaborate in an "intelligent" manner all together with a common general purpose. These platforms can be seen as a networks to connect citizens and ideas for social innovation, leveraging on collective intelligence and actions to address sustainability challenges. They exploit the emerging "network effect" by combining open online social media, distributed knowledge creation and data from real environments. Can be built systems where users will have an engaging and socially rewarding experience, but at the same time they will generate a stream of Geo-referenced images of mountains, feeding an existing mountain image analysis model. This paradigm fills perfectly into our idea since thanks to the collaboration of users is possible to populate and enrich the collection improving further analysis.

- *Game With A Purpose*: This line of work focuses on exploiting the billions of hours that people spend online playing with computer games to solve complex problems that involve human intelligence [5]. The emphasis is on embedding a problem solving task into an enjoyable user experience, which can be conducted by individual users or by groups. [4]. The gamification is a crucial part of our project that will be take into account closely and some ideas will be presented in the Chapter 6 about Future Enhancement.

1.2 Augmented Reality

Augmented reality (AR) is a live direct or indirect view of a physical, real-world environment whose elements are augmented (or supplemented) by computer-generated sensory input such as sound, video, graphics or GPS data. It is related to a more general concept called mediated reality, in which a view of reality is modified (possibly even diminished rather than augmented) by a computer. As a result, the technology functions by enhancing one's current perception of reality. By contrast, virtual reality replaces the real world with a simulated one [6]. Augmentation is conventionally in

real-time and in semantic context with environmental elements. With the help of advanced AR technology (e.g. adding object recognition) the information about the surrounding real world of the user becomes interactive and digitally manipulable. Artificial information about the environment and its objects can be overlaid on the real world. To obtain this result the AR technology is based on both hardware and software components. About software development the amount of works of algorithm solutions designated to object recognition is large. In specific the mountain tagging problem has been already amply faced. From the hardware point of view besides the typical ones, as processors (CPU and GPU) which computes the calculations and displays where we visualize the results merging reality and artificial contents, the main role is played by the sensors and input devices. In our case we adopted accelerometer, gyroscope, GPS and images from camera. Modern mobile computing devices like smartphones and tablet computers contain all these elements making them suitable AR platforms. The principal player is the camera, through which it's possible to acquire the surrounding world aspect, capturing the visible peaks and then crossing them with the on board sensors. In our work the choice to exploit the AR technique to display the results of the mountain peak detection aims to offer to the user an attractive and immersive experience.

1.3 The MountainWatch Project

The project aims at developing and evaluating MountainWatch, an environment protection mobile application based on augmented reality and backed by a social web platform. MountainWatch leverages an algorithm for peak detection in mountain images and transforms it into a Collective Awareness Application for citizen-driven environment monitoring. MountainWatch will run in the user's mobile phone. It is able to identify mountain peaks in user-generated photos in real time and then to enrich the photos with peak name tags. With MountainWatch users will have an engaging and socially rewarding experience, but at the same time they will generate a stream of geo-referenced images of mountains, feeding an existing mountain image analysis model. This stream will be made available as input to an existing model that performs mountain monitoring through digital terrestrial modeling and image processing. It estimates very important physical parameters such as the extension and depth of snow cover, snow water equivalent, glacier speed, and vegetation cover. Participation will be boosted by a social web platform that implements a competitive game based on achievements (e.g. missions to photograph a given mountain peak) and on the positive environmental impact of the user's activity. MountainWatch can help address a fundamental social challenge: environment monitoring through the automated analysis of user generated, low-cost visual content.

1.3.1 Environmental Monitoring System

The ultimate challenge of the MountainWatch is boosting social engagement for environmental monitoring. In a period of climate change and shrinking public investments in monitoring infrastructures, the need of a low cost analysis of environmental and ecological parameters is extremely important. A few methods are now proposed that try to virtualize the permanent measurement stations using mountain image analysis, using both ground and aerial images. These methods suffer from the absence or high cost of input data (e.g., satellite imagery, ground photos of a specific peak).

Most approaches require ad-hoc cameras installed in the mountain regions. These approaches are insufficient to produce and calibrate a really usable mountain environmental model. On the other side, publicly available mountain photos are available in great quantity in sites such as Twitter, Instagram, Flickr. The main difficulty that prevents using this content is the need of an automatic mountain peak identification tool. Mountain peak detection from casual images is a challenging task and the very few academic algorithms proposed recently tend to fail due to the the imprecision of the positioning data associated with the photo. A solution to this uncertainty could be applying the mountain peak identification procedure in real time when the user is taking the photo with his mobile device. This would improve significantly the accuracy of the results because the user would be encouraged to switch on the GPS sensor and thus provide precisely the GPS data and the direction of view of the camera.

MountainWatch will engage the user in a societal challenge of producing high quality images, by identifying peaks in real time and overlaying their name onto the user generated photo, producing a very attractive and aesthetically pleasant image, which the user will love to save and post to her social network neighborhood. As a side effect, the photograph and all its useful meta-data can be captured by MountainWatch (with the explicit consent of the user). MountainWatch will thus present itself to the user as a "cool" augmented reality camera for mountain peak detection and mountain photo processing. Behind the curtain, it will act as an active crowdsourcing interface for massive environmental data collection. The gathered images and their meta-data will be processed to feed data environmental monitoring systems, an example of which has been already implemented.

To boost engagement and virality, MountainWatch will be integrated with major social networks, so to let the user share her beautifully annotated pictures with friends, challenging them to download the app and do the same. This will turn MountainWatch into a tool for promoting social awareness about environmental problems, thanks to a competitive game model that measures the positive impact of the user's activity and her contribution to the mountain monitoring network.

1.3.2 Innovation

This work is innovative for two main aspects, from the social point of view and under the technical aspect. It can be defined as a Digital Social Innovation where a new type of social and collaborative cooperation is created in which innovators, users and communities collaborate using digital technologies to build together knowledge and solutions for a wide range of social needs and at a scale that was unimaginable before the rise of the Internet. The target of this proposed solution aims at the creation of a collective intelligence for monitoring the evolution of the mountain health status and for predicting important ecological and environmental phenomena, by a social collaboration (aggregation of user generated photos). Few studies about the performance of content analysis algorithms were already carried out but the mountain image data set collected can be tested only at very reduced scale. Here we propose instead to harness the love of people for mountains to gather a mass scale dataset, suitable for real life application scenarios, an objective surely impossible before the rise of the Internet and of social network applications.

A number of mountain peak augmented reality mobile applications already exist on the market. Among the most popular ones, we can cite Peak.ar, Peak Search - Alps, Panorama Tatr, Peak Scanner and ShowMeHills AR mountain peaks. All the listed applications are based on the same logic: given the camera GPS position and the three axis orientation (determined by the built-in gyroscope and accelerometer), the mountain peak position on the device screen is estimated by projecting the physical position of the peak (with known GPS and altitude) on the camera screen. Mountain peaks estimated in this way are not suitable for the environmental analysis, due to insufficient quality and precision. Our tests with these applications revealed that although the precision of the peak alignment could be sufficient for an entertainment purpose, it is not suitable for an automatic analysis for environmental purposes. In terms of quality, the mountain peaks near the photographer are always labeled, even when they are not visible.

Collecting photos in this way would introduce a huge amount of noise into the dataset compromising its usability, while providing a not-so-satisfactory user's experience, due to the irrelevant peak labels introduced in the user generated photo. The technical innovation of the proposed method resides in an image processing algorithm that aligns the picture viewed by the user to a synthesized rendered view of the terrain that should be seen from the users point of view, generated by a (publicly available) Digital Elevation Model. This alignment allows the application not only to estimate precisely where a mountain peak is located on the image, but also to detect whether the peak is visible or not (due to clouds, rain, obstacles etc.). This innovation greatly improves the user experience by positioning the peaks labels much more precisely and eliminating invisible peaks, and also makes the photo

and its meta-data usable for mountain environmental analysis.

1.3.3 Potential Impact

The application's main target group includes professional and amateur mountaineers, but also the common people who love to spend the day outdoor and take pictures of the trip with a nice mountain backdrop. The main virality factor is the high popularity and frequency of mountain trips among trekkers and hikers, and consequently the predictable high volumes of photographs that can be generated, as well as the very high enthusiasm and passion among both professionals and amateurs about mountain knowledge acquisition. Especially amateur mountaineers are generally incapable of identifying all the mountain peaks in the range they are visiting and would probably love an app that helps them acquire a deeper understanding of the region they are exploring. A similar effect is achieved, for example, by the many sky observation apps available in mainstream app stores, which count millions of downloads and have engaged also common people in sky watching. The plan to reach the target group is based on the engagement of the mountaineering associations. The approach to marketing and dissemination will be graded, so to scale from the local level, to the national level and finally to European and international level.
Currently, the International Climbing and Mountaineering Federation (UIAA) has a global presence on five continents with 80 member associations in 50 countries representing about 2.5 million people. The Club Aplino Italiano (CAI) alone, the major Italian alpine associations, counted 320.000 members in December 2011. A contact is already in place between Politecnico di Milano (through Prof Maria Brovelli and Prof Cesare Alippi) and the CAI. We plan to field test the app with the local chapters of CAI (in Lombardy, especially in the provinces of Brescia, Como and Lecco); after this phase, we will disclose the MountainWatch concepts to the CAI President and Steering Committee to obtain the official endorsement of the institution at the national level, as a vehicle for dissemination to all CAI members. Similar contacts will be pursued in other alpine European countries, starting from France, Switzerland and Austria, where academic contacts with research groups working on mountain environment problems have already been established. These groups will facilitate the connection with other national Alpine Clubs.
Other communication channels include environment-oriented media companies, such as LifeGate (www.lifegate.it), a national level, very popular media company, which has already well established links with Politecnico di Milano. The media relationship office of Politecnico di Milano has also very good connections with all the leading Italian television companies; we will exploit this connections to pursue the objective of obtaining a presentation of MountainWatch in one of the mainstream "green" Italian TV programmes,

which have an extremely broad audience.

1.4 Document Structure

In the next chapter I will discuss the related already existing projects presenting the current state of art about AR applications running on mobile devices and image processing algorithms about to environmental subject recognition.
In the third chapter the system is described deeply presenting in details all the components and matching algorithms adopted.
In the fourth chapter the implementation aspects are explained including all improvement technique developed.
In the fifth chapter the experimental study is described disclosing the performances of executed tests.
In the last chapter are exposed the final conclusions with the possible future improvements.

Chapter 2

Related Work and State of the Art

This work combines several different disciplines, starting from Digital Image Processing for the peak detection, to environment modeling able to perform the matching. More in relation with Social Computing there are Data Mining and Collective Intelligence to monitoring the mountains and Game With a Purpose (GWAP) System to stimulate customer interaction increasing satisfaction and employment. Augmented reality aims to display in a enchanting and innovative mode outcomes. The innovation consists of the fusion of all these systems available for mobile devices. In this chapter i will present some efficacy solutions for each single specific research field mentioned above.

2.1 Mountain Peak Detection

The real time peak detection is the key aspect of the entire work treated. Depending on platform type, mobile or classical desktop, where the system runs the approach is different. In the first case the on board sensors help to find out the mountain peak orientation and localization obtained respectively through the compass, accelerometer, gyroscope and the GPS signal but the available computational power is low. In the latter the platform can perform heavy calculations and the task changes according to the input photograph details without all the sensor information apart occasionally the GPS tag.

2.1.1 Sensor based Mobile Apps

In the Android application store "Google PlayStore" the availability and spread of such applications is high but until today the adopted approach

leads to an approximate result. In "Peak Finder Earth" application they start from the DEM (Digital Elevation Model) as reference to acquire the right position of mountains. Combining the geographical coordinates from GPS localization and the axis orientation by means of magnetometer, gyroscope and accelerometer and using them as input of basic equations of geometric transformation the approximate peak detection result is reached. In the next chapter i will explain deeply how the orientation is performed. Other systems as "Peak Scanner", "Peak.ar" or "Peak Search - Alps" all available on Google PlayStore are based on this same approach producing not so high profile results mostly due to low accuracy sensors, even on last recent mobile device models. The eye-catched error has not discouraged it usage as help for trekking beginners guaranteeing simple but profitable user experience.

2.1.2 Offline Desktop based System

In this different workplace both the input and output move away a lot from the above approach. In general the input pictures come from the Web, mainly social platform, the presence of geographical coordinates and camera sensor specifications can lead to different problem solving methods.
The tasks change according to the availability of GPS tag, changing the direction of computational effort owing to the different analyzed datasets. Also the objective is different: with the coordinate the problem consists of first Pose Estimation and then Mountain Identification phase. With the lack of localization estimate the challenge starts with a previous additional step: to analyze and to match the picture with a huge mountains dataset to recovery the correct geographical position.
An example of estimation of geographic position of a photograph is proposed by Hays and Efros [7] where the estimate of the position is performed with the analysis of the visual content of the image, a purely data-driven scene matching approach is applied to find out a geographic area whose the photograph belongs to.
A sort of inverse study is presented by Ramalingan et al. [8] exposing how to retrieve the position by means the of orientation. The authors describe a method to accurately estimate the global position of a moving car using an omnidirectional camera and untextured 3D city models. The idea of the algorithm is the same of ours: estimate the pose by matching the input image to a 3D model (city model in this case, elevation model in case of our work).
These and other similar works in such context have been done in order to face with Pose Estimation given the reference 3D model mainly in urban context rather then in mountains environment. The problem to source the urban 3D models has been overcome thanks to last years massive growth of 3D data of buildings and cities. In our context the data source come

from the terrain elevation data: the elevation data, presented usually as a geographical grid with the altitude for each point, can be easily seen as a 3D model, and the most interesting objects formed by these models are for sure mountains.
Fedorov et al. in [1] and [9] has been developed an automatic technique that, given as input a geo-tagged photograph, estimates its FOV (Field Of View) and the direction of the camera using a matching algorithm on the photograph edge maps and a rendered view of the mountain silhouettes that should be seen from the observer's point of view. The extraction algorithm then identifies the mountain peaks present in the photograph and their profiles. The outcomes are really interesting: with the best parameters configuration the algorithm has correctly estimated the orientation of 64.2%. The alignment performance depends mostly on three agents: the camera sensor adopted where the cellular phone specification about the FOV are less accurate than classical camera leading to incorrect scaling estimate. The presence of the clouds influence the edge extraction introducing noise and decreasing alignment score. Furthermore as third the composition of skyline influences the correct matching according to the landscape: with only mountains and terrain the obtained results are significantly better rather then scenes with foreign objects. The thesis proposal with respect to SnowWatch project is oriented towards mobile platform to perform the entire matching process on-the-fly displaying almost instantaneously results in a real-time way through augmented reality system. The adaptation and re-engineering to different platform, computing power and context use ranges from different algorithm adoptions, exploiting different input information and showing outcomes in a graceful manner.
The project exposed by Baboud et al. [10] is the another pertinent work and landmark for the purpose of this project. Given an input geotagged photograph, introduces the matching algorithm for correct overlap identification between the photograph boundaries and those of the virtually generated panorama (based on elevation datasets) that should be seen by the observer placed in the geographical point where the photograph has been taken from.
Naval et al. [11] also describe an interesting solution to perform the position estimation of mountain image: the distinctive trait derives from the choice of do not work with the complete picture edges of the image but only with the skyline extracted using a neural network. The position and the orientation is then computed by nonlinear least squares.
Excellent outcomes are reach by Baatz et al. [12] with a proposal of an algorithm that given a photograph, estimates its position and orientation on large scale elevation models (in case of the article 14 2 a region of 40000 km was used, but in theory the technique can be applied to position estimation on a world scale). The algorithm exploits the shape information across the skyline and searches for similarly shaped configurations in the large scale database. Evaluation was performed on a data set of photographs with

manually verified GPS tags or given location, and in the best implementation 88% of the photographs were localized correctly.

2.2 Collective Intelligence

The actual environment monitoring is carried out by means of expensive and complex systems as e.g. permanent measurement stations using images collections obtained acquired from ground and aerial images with high costs also due to ad hoc and rare specific cameras installed in the mountain regions. Exploiting the Collective Intelligence is possible to build a smart, widespread and relatively inexpensive environment monitoring system reaching affordable results and contemporaneously transforming the proposal to Collective Awareness Application.
These exciting findings directly connected to the huge amount of social network sharings brings to create an innovative system to take advantage of this simply accessible but enormous quantity and by now high quality data. In fact processing those public uploaded photos is possible to gather precious and valid information otherwise unexplored and unexploited.

2.2.1 Game with a purpose

Location-based services (LBS) gain a lot of attraction recently. However, to collect necessary and accurate information to support LBS is always manpower-consuming, and may be also difficult for machine computation. Human Computation can therefore be adopted to attack the problems. These problems which are usually difficult for machines to solve can be easily completed by human. We can embed our purpose into a designated game. In this way, players will automatically achieve our purpose while playing the games.
The popularity acquired from location-based apps, like Foursquare where people tends to share information of their location with their friends, incentivized the employment of this data collection to gather information about urban environment leading to further interesting studies about people social behaviors. This trend of Human Computation connected with a social mobile GWAP is presented in Urbanopoly [13] project: exploiting a qualitative and quantitative approach is possible to discover how the physical presence in the urban environment together with location based technologies can prove a valuable contribution to geo-spatial information, and how the direct experience and human sensing can play an important role in solving these tasks related to the physical space. The stressed key aspects are the introduction of set of several tasks to be performed, generalizing the approach to achieve multiple purposes; assuring a long term engagement of the player increasing the Average Life Play (ALP) and exploiting the social dimension by integrating Facebook within the gameplay.

2.3 Augmented Reality

In representing results the choice of Augmented Reality (AR) allows the users to be engaged, being more interactive and immediate. This approach can guarantee social acceptance due to the subtle, discrete and unobtrusive experience given the size device, moreover the user is able to interact in a natural and instinctive way. The AR systems can bring a better solution to some areas, a cheaper solution to others, or simply create a new service. The covered fields of AR applications are several and quite different, covering a large range of interests from advertising and commercials, to entertainment and education and to medical applications.

2.3.1 Augmented Reality Systems

Taking into account the educational and cultural aspects in Huang et al [14] this system reconstruct ancient ruins where is possible to admire the Yuanmingyuan magnificent royal garden otherwise impossible to be perceived. In this specific case the 3D superimposition comes from an ad hoc optoelectronic imaging system strictly designed and adopted for this scope. In a museum or cultural exhibitions in the case of smart-phone interface based application can be reached efficient communication with the user through multimedia presentations, natural and intuitive technique and low maintenance and acquisition costs for the museum operator's presentation technology.
Around the most profitable field in AR gaming applications besides the obvious entertainment aspects acquires a relative importance the ability to introduce animations can not only add excitement to a game, but it can also serve a learning purpose with, for example, indication to help players learn the game or know when they are making an invalid move.
Changing sector most of the medical applications deal with image guided and robot-assisted surgery. As a result, significant research has been made to incorporate AR with medical imaging and instruments incorporating the physicians intuitive abilities. Significant breakthrough has been provided by the use of diverse types of medical imaging and instruments, such as video images recorded by an endoscopic camera device presented on a monitor viewing the operating site inside the patient. Bichlmeier et al. [15] introduced an AR system for viewing through the "real" skin onto virtual anatomy using polygonal surface models for real time visualization. The authors also integrated the use of navigated surgical tools to augment the physicians view inside the human body during surgery. Teleoperated robot-assisted surgery provide the surgeons with additional advantages over minimally invasive surgery with improved precision, dexterity, and visualization. However, implementing direct haptic feedback has been limited by sensing and control technology and thus is restricting the surgeon's natural skills.

The lack of haptic feedback has been proved to affect the performance of several surgical operations. In [16] the authors propose a method of sensory substitution that provides an intuitive form of haptic feedback to the user. The need to reduce surgical operations is not the only one to depend upon seeing medical imaging data on the patient in real time; the necessity to improve medical diagnosis also relies on it. In general the use of AR in the medical field to provide better solutions to current problems than already existing solutions is infinite.

2.3.2 Augmented Reality On Mobile Devices

The nowadays availability of Augmented Application on mobile devices is mainly due to the large step forward of the computational power of modern portable platform and the parallel rise of social web platforms. Thanks to the first aspect is now possible to perform heavy calculations exploiting accurate and precise algorithms and approaches obtaining never seen results. The Porzi et al. [6] proposal is the most suitable project with respect to this research sector. The system relies on a diverse approach for robust registration between the real scene and a synthetic representation of the world, i.e. profiles automatically generated from Digital Elevation Models (DEM). This method starts with a rough estimate of the orientation and position of the device that is computed by processing data from on board sensors, i.e. GPS, magnetometer, gyroscope and accelerometer. This estimate is then refined by means of a different alignment algorithm that exploits visual information. The adopted algorithm matches edges extracted from the given image against synthetic profiles, guided by a scoring function supporting the best alignment. With respect to [10] Porzi et al. adopts a simpler and faster scoring function and devise a learning-based edge detection approach. This prevents the occurrence of many spurious edges, thus lightening the subsequent alignment process. Once the photo-to-world registration procedure is completed, virtual content is rendered and overlaid on the real scene. As in Ozuysal et al. [17] they exploit a simple and efficient way to detect pixels corresponding to mountain profiles, according to visual features extracted from their neighborhood. Standard edge detectors, such as Canny [18] and Compass [19], treat image edges equally regardless of their context. However, the edges of a specific object (i.e. mountains) have the characteristic local color or texture of that object on one side. A learning-based approach, oppositely to standard algorithms, is able to capture this information, filtering out spurious edges corresponding to other objects (e.g. man made structures). The outcomes of the experimental evaluation confirmed accurate registrations unite to almost computationally efficient algorithm.
Similar approach with a different pertinence the Bottari et al. [20] presents an augmented reality application that offers personalized and localized recommendation of Point of Interest (POI) based on temporally weighted opin-

ions of social media community. This service continuously analyzes the social media streams understanding how the social medias users collectively perceive the POIs in a given area.
Finally Wikitude Driven project is a real time augmented application available for Android and iOS platform suitable for navigation purpose: superimpose onto the image street the correct path to be follow to reach the destination.

Chapter 3

Problema Statement and Proposed Approach

In this Chapter I will present the adopted approach to build our system supplying the reasons of our choices. I will describe all the phases composing the entire process and the possible areas of applications where our solution can be adopted.

3.1 The Motivations

There are many motivations leading to a development of this new innovative application. The reasons have to be searched in several aspects. First, the society evolves and becomes more and more technological. People have more social predisposition, inclination and pays attention about ecology and safeguard of the nature.
The possibility of photo sharing on the principal social networks brings useful content as the geographical position and information about the surrounding environment, in specific mountains. These resources could reveal important for nature monitoring and safeguard of the environment.
Particular importance derives from the excellent outcomes reached in such research fields as mountain peak detection inside professional photographs. The photos are taken generally with reflex cameras and detection exploits robust and heavy algorithms achieving high level precision.
Easily to perceive is also the elevate request and relative spread of mobile social applications designed for mountain context as the localization of surrounding peaks. These applications typically still make errors with a visible imprecision; often the user experience is not so satisfactory given the lack of efficient and precise content analysis.
Increase of the mobile systems computing power plays also an important role. These devices are almost comparable with desktop systems and photo-

graphic industry for mobile platform has really equalize the level of nowadays digital cameras.

3.2 The Proposal

Given all these aspects we decided to create a new type of user experience: wherever in presence of mountains, typically during a mountain trip, when we find in front of an interesting landscape with as subject a majestic visible peak a beginner trekker or climber, or simply interested observer, is able to recognize which mountain he is looking at. He can obtains some interesting information about the peak, the whole reachable instantaneously simply using his smartphone through an augmented reality experience. The information about the peak is overlapped on the current preview frame, obtained thanks a long and deep process of image content analysis, augmenting precision step by step exploiting cross platform information as on board sensors and external web services. The smartness and goodness of adopted algorithms giving also the possibility to detect if the current peak in front of the user is really visible or maybe some obstacle shadow the view as trees, buildings or whatever not belong to nature world. A key aspect is the possibility to augment the reached precision step by step, visualizing almost in real time the boost without waiting the end of the whole process, given the heaviness of the whole system.
To underline the step by step progression of process the marker on the camera preview representing the peak position changes color according to the current phase. This feature is adopted for each forward step, passing from "red" color to inspire wrong position to yellow of the first step to underline an improvement. The the light blue explain the reached good estimate of the global alignment, adding some shades to green reaching the final stages.
The application, keeping always active the camera preview during the execution, gives to the user the possibility to focus the interested areas visualizing it directly on the display. To help there is an over layer representing the compass indicating the the cardinal directions. There are also some functionality to configure according to the user preferences; set the maximum distance within which the mountains can be marked as visible. During all the execution is possible to take photo to share and show the result of the peak recognition immediately or later. The advanced mountain updater can be also activated or deactivated corresponding to the image processing.

3.3 The Application

The general overview of the phases that compose the process is shown in Figure 3.1.

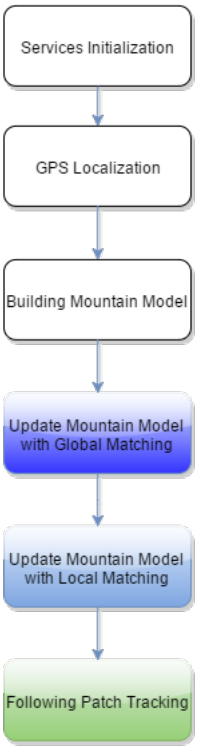

Figure 3.1: Application Schema of the phases composing the entire process. The color choice helps to understand the evolution of process: the first rough sensor peak estimate (white), the alignment based on first global content analysis (blue) and on the second local content analysis (light blue) and the ending phase of tracking based on the following patch matching (green).

After the Services Initialization and GPS Localization the Mountain Model is built. From now on the visualization of peak names is based only on sensor keeping track of the device orientation. Also after the updates on the model coming from the Global Matching and Local matching the labels in the camera preview are based on the orientation. In the final stage when the Following Patch phase starts the mountain names are not anymore estimated by sensors. Given the last peak positions updated during the Local Matching the patches around the visible peaks are extracted. Now the displayed labels are the result of the matching of the stored patches inside the current frame. Now I will present deeply all the local processes and external active services during the entire application lifetime.

3.3.1 Services initialization

First of all a small hint is given at each application boot: a simple gesture explained through a brief animated Graphics Interchange Format (GIF) image to encourage a sort of sensor calibration moving the device drawing the

number "eight" as shown in Figure 3.2. This helps the magnetometer to reset and restart the estimate of the cardinal directions.

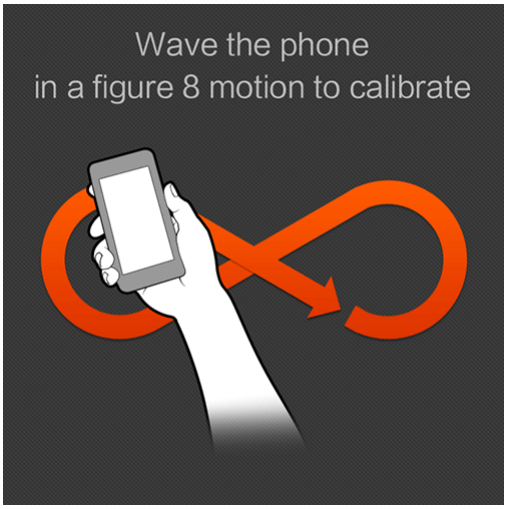

Figure 3.2: Compass Calibration: how to move the device to help the sensor calibration at the application boot.

Another little feature is inserted in the application boot: to perform the image processing we decide to adopt an external multi platform library, OpenCV exploiting the java interface for android and we have to check the presence of these modules. In the Chapter 4 I will present deeply the motivations about the adoption of this strategy.
After the installation of MountainWatch, in future available simply in the official store, at the first start an immediate check controls if the library is already present in the OS. In a positive ending the user remains totally transparent, otherwise a pop-up is show to present the possibility to install the required application directly from the store simply clicking on "yes, i want to install".
As suggested before to progressively acquire more and more precision almost all available on board sensors are adopted. As before the OpenCV external app they cannot be avoided because otherwise it's impossible to start entire process. The OS gives instant access to all the needed instrument, helping the user to set properly everything. So at start together with the linked reference check a simple pop-up making conscious the user about the necessity of switch-on GPS and internet connectivity. The accelerometers and gyroscope don't need any check and work in background managed directly by the OS routines.
The requirements are perfectly aligned with nowadays released applications. The mobile Internet connectivity is available almost on the whole territory and the percentage of smartphone user without any data connectivity is very low. On this point it will then analyzed deeply in the Chapter "Future

improvements about possible enhancement" where a possible solution only local based is presented.

3.3.2 GPS Localization

After these initialization tasks, the GPS localization is the very first starting point. This gives the possibility to localize and so understand the actual mountain chain present in that area. Acquired the information about latitude, longitude and altitude the detection about which peak is really visible from the camera can start.
The GPS tag precision achieved is suitable for our target, even if the Egnos system has been taking into account. This solution can lead to a precision of the final GPS tag improved from 8-10 meters to estimates under the 7-5 meters. The enhancement it's relevant but contextualizing in our situation the imprecision coming from the other sources involve a no real improvement. This no real enhancement derives from the 2D projection of digital elevation model received from the external service udeuschle.de: even if the final localization becomes more precise the subject still remains far from the viewer and some meters doesn't affect a lot the final projection. In Figure 3.3 a typical Panorama 2D reconstruction from this external service is shown. In future developing our internal service with higher precision can be taken into account more reasonably.
The concrete problem comes from the precision of the Digital Elevation Model (DEM). The DEM is the representation of altitude distribution of a territory. The digital model is generally produced in a raster format associating at each pixel the corresponding absolute height. If the initial model represents not so faithfully the real mountain model all the subsequent passages will have less probability to augment the final peak estimates.

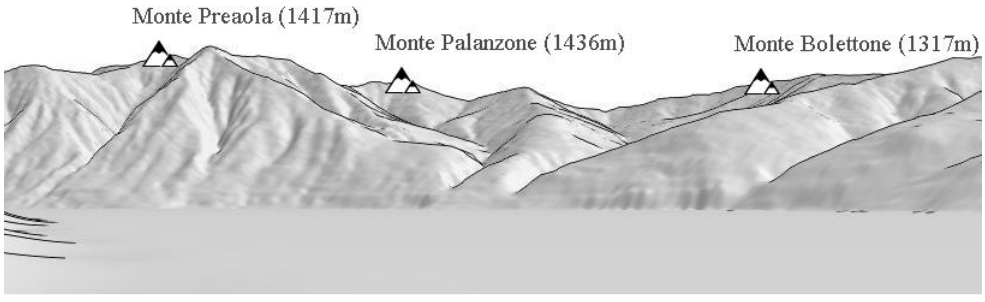

Figure 3.3: Extracted Patch from Panorama 2D with mountain markers.

Once all the checks are gone, now it's possible to send the request to our

internal services to acquire all the necessary data to finally start the peak position recognition.

3.3.3 Building the Mountain Model and Sensor Matching

Once recovered the location a request containing the outcome GPS tag is sent to our internal server. Server side a request to an external service, udeuschle.de, is sent. This service receiving as input the location and some configuration parameters returns the projection the 3D model of mountains, recovered from the open source DEM, to a 2D plane representing the 360 view of the user, the Panorama 2D. Together with the Panorama 2D server side we acquire information about the mountain, as the name, the altitude and the peak positions inside the received panorama image. An analysis on the contained is executed to understand which peaks can be really visible: only the nearer one are preferred in order to avoid confusion due to too many overlapping peak markers. When a request from the mobile app has the same locations coordinates a cached version is available and is immediately send in the response, otherwise a new request to the external service is sent. When in the application Panorama 2D and all the related details are received the first representation of peak estimation starts.

In Figure 3.4 is shown the sequence of working frames starting from the initial data downloaded from the network and finishing to the pixel coordinates on the mobile device display. There are four working frames: the Panorama 3D Frame, the Panorama 2D Frame, the Camera 3D Frame and the Camera 2D Frame.

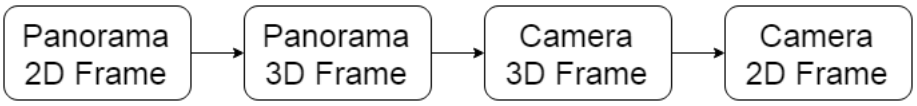

Figure 3.4: The peaks are received from the network are in pixel coordinates in the Panorama 2D Frame. To display the peaks in the camera preview three transformations are applied passing from Panorama 3D Frame, Camera 3D Frame and Camera 2D Frame.

From Panorama 2D Frame to Panorama 3D Frame

The 2D frames correspond to panorama and image pictures, as presented before. They coincide to the lateral surfaces of cylinders as shown in Figure 3.5. The 3D frames represent the concentric cylinder systems as described in Figure 3.6.

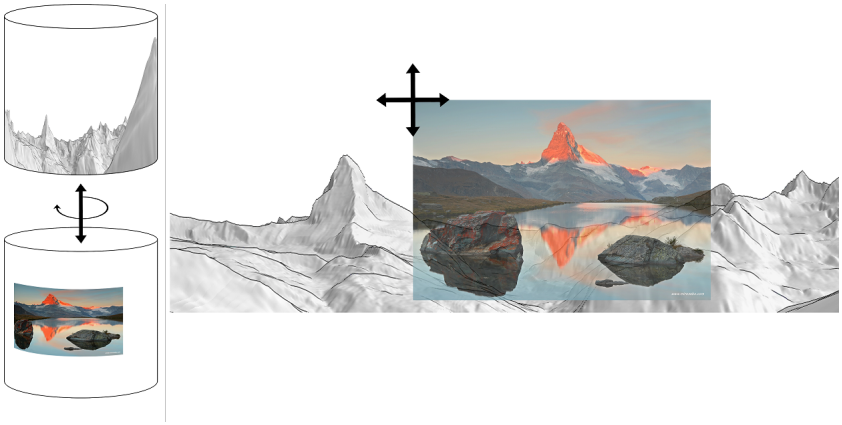

Figure 3.5: The overlapping of Panorama 2D and Camera 2D adapted from [1]

First we extract from the received file the position of the peaks inside the downloaded panorama in pixel coordinate, the Panorama 2D. Then we transform this pixel coordinate to Panorama 3D space coordinate.

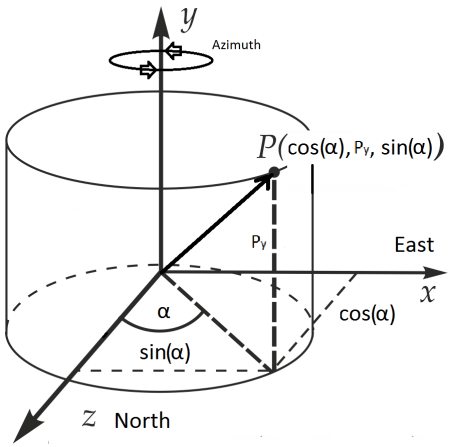

Figure 3.6: Cylindrical Coordinate System applied to rotational axis and geographical directions of testing mobile device.

The peak pixel coordinates (mx, my) in the 2D panorama frame received by the JSON file have to be transformed into a vector to representing mountain in Panorama 3D frame. To apply trigonometric rules we have to pass from the Original Panorama System to the Calculation System as presented in Figure 3.7.

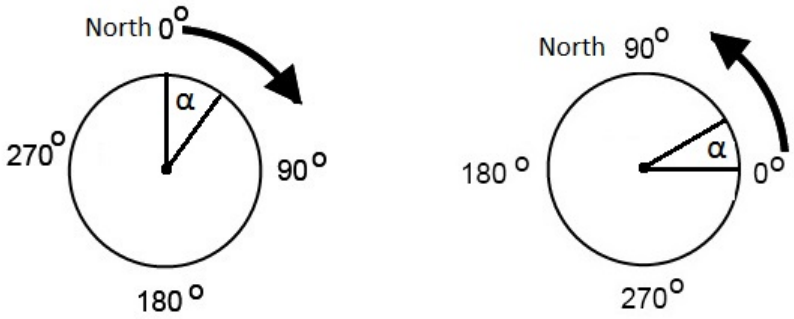

Figure 3.7: The Original Panorama 3D Frame on the left and Calculation System on the right.

The original peak in the Panorama 3D is 3.1

$$peak_o = (x, y, z) = (cos(\alpha_o), p_y, sin(\alpha_o)) \tag{3.1}$$

where the angle representing the direction of the peak is 3.2.

$$\alpha_o = -s_f mx + \frac{\pi}{2} \tag{3.2}$$

We have to change system so $\Pi/2$ is added and the direction of rotation is inverted. The scaling factor s_f 3.3 represents the quantity of *degreeperpixel* of panorama picture along both direction, vertical and horizontal.

$$\frac{HFov_{pan}}{W_{pan}} = s_f * \frac{HFov_{cam}}{W_{cam}} \tag{3.3}$$

$HFov_{pan}$, W_{pan}, $HFov_{cam}$ and W_{cam} are respectively the horizontal Panorama Fov, the vertical Panorama Fov, the horizontal Camera Fov and the vertical Camera Fov. The Fov is the Field of view. 3.4 represents the height of the peak in the Panorama 3D, where R_h is the panorama height.

$$p_y = \frac{\frac{R_h}{2} - my}{\frac{R_h}{2}} \tag{3.4}$$

From now on the peak will be represented always in Panorama 3D frame, so the future updates will fix the vector instead of the pixel coordinates.

From Panorama 3D Frame to Camera 3D Frame

Now applying the Transformation Matrix to the original peak vector the peak in Camera 3D frame is found. The peak in Camera 3D frame is the peak saw from the viewer point of view.
The Transformation matrix is the product of the Viewing Matrix and the Projection Matrix 3.5. The panorama picture received is already the projection of the mountain model so the projection matrix is an Orthographic Projection, and identity matrix.

$$M_{transf} = M_{viewing} * M_{ortho} \tag{3.5}$$

The Viewing Matrix is obtained by the reading of the orientation sensor so the azimuth, pitch and roll. The Viewing Matrix is the result of the product of four transformations 3.6.

$$M_{viewing} = ((M_{translation} * M_{rot_{azimuth}}) * M_{rot_{pitch}}) * M_{rot_{roll}} \tag{3.6}$$

Given the angles of azimuth, pitch and roll respectively α, β and γ and the camera eye vector 3.7

$$Eye = \begin{pmatrix} e_x \\ e_y \\ e_z \end{pmatrix} \tag{3.7}$$

the transformation matrices are 3.8, 3.9, 3.10 and 3.11.

$$M_{translation} = \begin{pmatrix} 1 & 0 & 0 & e_x \\ 0 & 1 & 0 & e_y \\ 0 & 0 & 1 & e_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$M_{rot_{azimuth}} = \begin{pmatrix} cos(\alpha) & 0 & sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -sin(\alpha) & 0 & cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

$$M_{rot_{pitch}} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\beta) & -sin(\beta)0 & 0 \\ 0 & sin(\beta) & cos(\beta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

$$M_{rot_{roll}} = \begin{pmatrix} cos(\gamma) & -sin(\gamma) & 0 & 0 \\ sin(\gamma) & cos(\beta) & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

We apply the Transformation Matrix to find the $peak_f$ in the Camera 3D 3.12

$$peak_f = M_{transf} * peak_o \tag{3.12}$$

From Camera 3D Frame to Camera 2D Frame

The final step is the projection of the peak from the Camera 3D to the Camera 2D, so the current frame visible in the camera preview by the user. First the horizontal and vertical scaling factors along x and y are found 3.13.

$$(k_x, k_y) = (cos(\frac{\pi}{2} - \frac{HFov_{cam}}{2}), \frac{VFov_{cam}}{VFov_{pan}}) \tag{3.13}$$

Then they are applied to the resulting peak 3.14.

$$peak_f = (\frac{peak_f(x)}{k_x}, \frac{peak_f(y)}{k_y}) \tag{3.14}$$

Finally we find the peak in pixel coordinate in Camera 2D 3.15.

$$peak_{f_{clip}} = ((peak_f(x) + 1) * (\frac{W_{cam}}{2}), (1 - peak_f(y)) * (\frac{H_{cam}}{2})) \tag{3.15}$$

where W_{cam} and H_{cam} are the width and height of the camera frame.

The immediate result showed is the estimate based only on the on board sensors. Given the 3D coordinate of the peaks, at each frame the Transformation Matrix updated according to the current orientation of the device is applied to recover the respectively Camera 3D and then pixel coordinates.

A little maker representing a mountain is placed on the preview as shown in Figure 3.8, with an inclined label specifying the maximum altitude of the peak and its name. All the label with the names are oblique to be more readable in case of close peaks horizontally.

Figure 3.8: Compass Matching Result: the white marker represents the peak estimate only based on embedded sensors. The green marker is the estimate obtained from the groundtruth.

The result is a camera preview focused on the peak with an over layer containing the essential information about the in front of mountains. Those peaks fall outside the field of view in that specific moment are not visualized, until the user start to focus on them. Meanwhile the sensor matching goes ahead, there is the possibility to capture the visualized detection taking a shutter comprehensive of the over layer. Each time the user is moving and reach a new position where the distance with respect to the initial position at the boot app is higher than a certain threshold and a new GPS tag is revealed: if it corresponds to a new panorama projection then it has to be downloaded and the entire process has to restart.

3.3.4 Global Matching

When the first camera preview frame is available together with the panorama picture and the mountain coordinates in Panorama 3D, the image processing can start.
The adopted algorithms are taken from the SnowWatch project and now the differences due to the different context will be explained. In the first place thanks to the estimate of compass the range of the panorama to be taken into account is by far reduced. From the initial view of 360 degrees a slice of 75 degree is extracted. The center of the picture in the Camera 2D is reported into the Panorama 3D with the inverse transformation matrix. Reprojecting this 3D point vector on the Panorama 2D we obtain

the estimation of the center of the current frame onto the Panorama 2D. A panorama patch on the frame center projection is extracted, representing a larger slice of panorama at which the camera is pointing to. The horizontal Fov is about 60 degrees, the slice is 75 degrees. The frame is scaled adopting a simplified formula: with respect to the professional camera the OS directly offer access to the vertical and horizontal Fov avoiding the calculations necessary given the specs of reflex cameras, as focal length and sensor size. Before to apply the Vector Cross Correlation (VCC) several steps are performed. The edges extraction is performed adopting different horizontal and vertical filters to both the picture, panorama and frame. The impact on computational effort is otherwise unacceptable: there are other better edge detectors, as compass [21]. They are largely more precise and suitable for this application field but need too much computational power to be executed locally on portable device. To maintain the application flowing and reactive the Sobel filters 3.9 reach good trade off among computational effort, performance and outcomes.

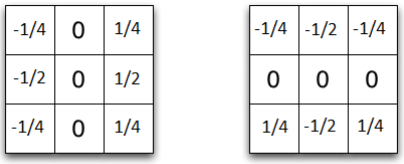

Figure 3.9: Sobel filters adopted.

About the thresholding, the threshold value has been obtained from a dataset of hundreds of shutters in different conditions, with and without fogs, at high altitude in a snow context and on-the-lake banks. The skyline detection follows the same idea, a simplified version has been developed to highlight the upper edges given the bigger probability to find the mountain profile 3.16.

$$img(r,c) = \begin{cases} 0 & \text{if } img(r,c) <= th \\ img(r,c) * k_{sky} & \text{if } img(r,c) > th \end{cases} \tag{3.16}$$

where rows is the frame height. The edge filtering approach analyze the picture column by column: starting from the top pixels towards the bottom, only pixel values higher then a predefined threshold th are multiplied by the factor

$$k_{sky} = \frac{rows - r}{rows}$$

This filtering helps in several context as cloudy shutters, photographs with different mountains one in front each other with low visibility or a lot of annoying objects.

Finally the VCC is performed giving as output the most probable offset to be applied to find the better overlapping of the frame onto the panorama. The output matrix is weighted with a Gaussian Kernel; this kernel function is centered in the middle of the frame with the scope to highlight the edges that belong to the center of the frame projected onto the panorama estimated thought the compass.

Figure 3.10: Global Matching Result: the blue marker represents the peak estimate after the Global Matching Algorithm. The green marker is the estimate obtained from the groundtruth.

This offset has to be applied to all the peaks, first to the Panorama 2D system updating the position directly on the panorama. Then the transformation to the corresponding 3D vector point in the panorama 3D coordinate system is found exactly in the same way as before.

Once this entire process has finished all the peaks in Camera 3D systems are updated and ready to be visualized in the consecutive frames as shown in Figure 3.10. The user can take advantage simply remaining in the same position or also walking around (he cannot take trip on car or other vehicle because the GPS detects new position and the process has to restart) still staring the surrounding mountains. All mountains have been updated because this first alignment has to be intended global, as a correction of the compass error. Also those doesn't fall in the Fov of the current frame taken into account during the alignment are updated. The previous estimate is hidden but the vector point indicating the original mountain position into the Panorama 3D system is not canceled, simply a new value is added. The reason is simple: when the user stop the advanced alignment or the application is paused the original estimate is still require to redo the entire process.

3.3.5 Local Matching

To improve the outcomes obtained from the Global Matching a different approach is taken into account. Due to a not enough precise DEM the corresponding panorama picture received is not so similar to the real mountain profile, the optical distortion can be a possible cause of projection imprecision. Sometime some parts are perfectly overlapped and other are not so well stretched and maybe an area of the frame is precisely aligned while the opposite region remains consequently quite imprecise. To avoid this situation another alignment matching can be applied in a different manner: local patch alignment.

Figure 3.11: Local Matching Result: the light blue marker represents the peak estimate after the Local Matching Algorithm. The green marker is the estimate obtained from the groundtruth.

For each peak visible in the image, and only those peaks visible in the user Fov, only a small patch around it is extracted. In this way different patches belong to the same frame can have a different alignment offset overcoming the panorama imperfections. So all the peaks are not aligned in one time all together as before but singularly each offset is calculated. Moreover only the peaks that are quite far from the edge. The reason is simple: if the mountain remains too close the border it means that it's often not entirely visible. To perform a good matching the patch needs to contain at least the central profile. Also the same reasoning is applied to the corresponding panorama, where a small region is extracted with a similar dimension patch as for the current frame. The peaks have not been processed will be take in the next iteration when the user focus the camera in their direction. If only one of the two patches cannot be extracted the alignment for this specific peak is jumped. In the next iteration a new check is executed:if the peak has at least a minimum distance from the border then the alignment can start. The iterations continues until all the mountains has been processed The process, a part the reduced dimensions, is the same as before, so edge extraction,

edge filtering and skyline detection. Reducing dimension means reducing computational effort and the resulting effect is more smooth, reactive and pleasant. The VCC is found and the Gaussian kernel as before is applied to the result.

To understand if a peak is really visible and not shadowed from obstacles a Potential Edge is estimated. It represents a threshold obtained from an empirical approach: only the values of the matrix higher then the Potential Edge are taken into account as new offset for the mountain updater.

At the end once the offset is available one by one all the vector 3D peaks are individually aligned, updating the value of the global alignment as shown in 3.11. This method give the possibility to increase precision working on local details around the peaks. As soon as the peak vector are updated the further frame will show the marker with a new color and with a less degree error.

3.3.6 Tracking

After the double matching phase is completed the mountains coordinates onto the image represent the best superimposition. From now on the target of matching algorithm changes: we do not have anymore need to align panorama and the updated 3D positional vector has taken as the best approximation. This stage is named tracking because now start the "following stage" where the outcome positions of the peak obtained from the Local Matching is researched in the subsequent frames. This tracking phase is performed locally adopting a similar approach of Local Matching: for each peak is extracted a surrounding patch representing the object to be tracked in the future frames.

The patch will not be updated: the future frames will search for the original patch extracted as the local alignment has finish for that peak. The Peak Tracking can be seen as a Template Matching, a technique for finding areas of an image (the current frame) that matches a template image (peak patch). Two primary components are necessary: the Source image I, the image in which we expect to find a match to the template image, in this case the current frame, and the Template image T, the peak patch image which will be compared to the template image. The goal is to detect the highest matching area.

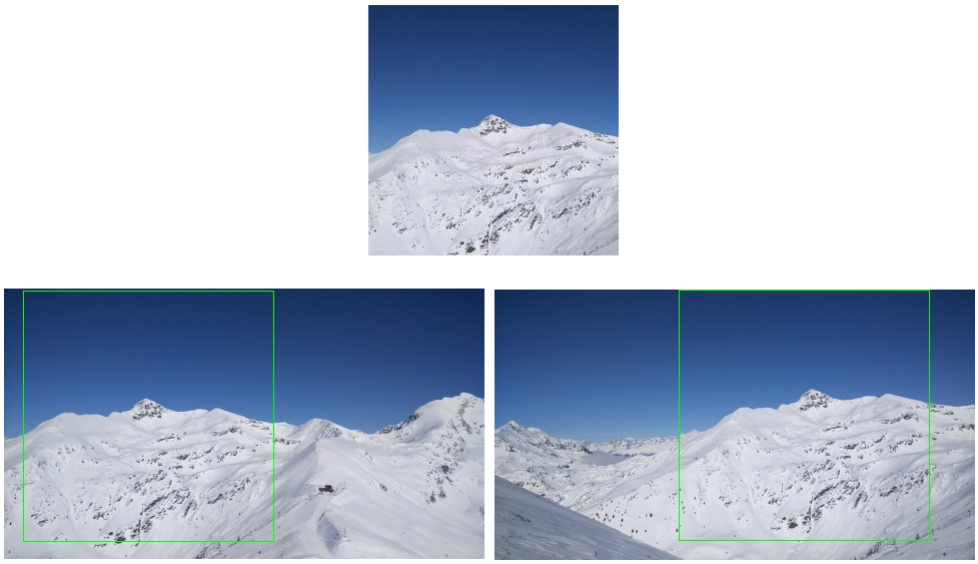

Figure 3.12: A patch retrieved in two different frames during the same sequence.

To identify the matching area, we have to compare the template image against the source image by sliding it: by sliding, we mean moving the patch one pixel at a time (left to right, up to down). At each location, a metric is calculated so it represents how is good or bad the match at that location is (or how similar the patch is to that particular area of the source image).For each location of T over I, you store the metric in the result matrix (R). Each location (x,y) in R contains the match metric: the image above is the result R of sliding the patch with a metric Normalized Correlation. The brightest locations indicate the highest matches. In practice, we use the function minMaxLoc to locate the highest value (or lower, depending of the type of matching method) in the R matrix.
Thanks to the adopted external libraries Opencv the performance of Normalized Correlation built in function gives the chance to display the output in real-time, bringing the experience of Augmented Reality: the peaks will be updated almost in Real Time while the user can rotate or simply move the camera looking all the mountain chain. In Figure 3.12 is presented the result of Tracking.

3.4 Possible Areas of Application

There are several application areas where this system can be exploited. This application is suitable for trekkers and mountain lovers to help the orientation and localization. The recognition of the surrounding peaks is displayed in a captivating experience. Another interesting aspect is the possibility to

evaluate of the sensor precision. The first phase of Sensor Matching can be possible method to analyze the compass precision of accelerometer and gyroscope estimating the error in a typical situation of mountain context. As already mentioned the environmental protection and monitoring is not the secondary one and now a particular proposal will be proposed.

Chapter 4

Implementation Details

In this chapter the implementation aspects of the approach proposed in the previous chapter are presented, from the general description of sensor functioning, to the main developed components and detailed presentation of the classes and methods created. The adopted programming language to develop application for Android is Java.

4.1 Adopted Sensors and Functioning

Inside almost all smartphones the vendors insert several types of sensors ever more precise. It's very common finds sensors as the gyroscope, the magnetometer and the sensor to receive the USAs Global Positioning System (GPS). A lot of applications exploit sensors to build several gaming, navigation or augmented reality app. This sensor integration gives to the developers new possibilities to enrich user experience in a more interactive and creative way. The available API gives to developers simple instruments and easy high level access. The result looking at the Play Store, the Official Android market of Google, are really impressive.
Now we analyze the orientation sensors, the gyroscope, the accelerometer and the magnetometer. As long the precision is related to the device orientation of the device the outcomes are optimal. When the scope changes direction towards the estimate of the cardinal directions the results are quite different. Depending on the goal in our mind the sensor can be or not suitable; anyway the estimation is sufficient for the classical use of smartphone. In fact the imprecision gives us the idea to correct it exploiting digital image. In fact with infallible orientation system the correction thanks to the content analysis it would be totally meaningless. The errors are present and perceivable and they are unstable over the time. The smartphones adopt concretely better and better sensors but they still remain not comparable to ad hoc and specific products. I carried on several practical tests, also

personally, in different situations and conditions explained in the papers [22] and [23]. As explained in these studies the sensors suffer terribly from external agents, mostly due magnet fields, without the possibility to directly control and eventually intervene on it.

Now I will present in details which ones we adopt, how they work and then I will show the related problems.

The device used for testing is OnePlus One showed in Figure 4.1, high-end smartphone available from march 2014. On board there is the embedded Qualcomm Izat GNSS WTR1625L chipset; it's embedded into the Qualcomm Snapdragon 801 together with the cpu, gpu and other modules. It supports a GPS, GLONASS and BEIDU satellite navigation systems.

Figure 4.1: OnePlus One, the device adopted to test and analyze the application.

The accelerometer and the gyroscope models are respectively the LIS3DH and the L3GD20 both produced by Microelectrics; instead the magnetometer is the AK8963 produced by AKM. The Specifications are shown in Table 4.1, 4.2 and 4.3 .

Model	LIS33DH
Vendor	Microelectrics
Range	39.226593 m/s
Resolution	0.019607544 m/s
Power	0.011 mA
Delay	8333 s

Table 4.1: OnePlus One Accelerometer Specifications.

Model	L3GD20
Vendor	Microelectrics
Range	34.906586 rad/s
Resolution	0.0012207031 rad/s
Power	6.1 mA
Delay	5000 s

Table 4.2: OnePlus One Gyroscope Specifications

Model	AK8963
Vendor	AKM
Range	4911.9995 T
Resolution	0.14953613 T
Power	5.0 mA
Delay	16666 s

Table 4.3: OnePlus One Magnetometer Specifications

The magnetometer detects the terrestrial magnetism along three axis, the x,y and z axis. Combining its measures with the output of the other two the north direction can be estimated. It's an electronic compass integrated circuit with high sensitive Hall sensor technology. The main noising agent is the external magnetic field in the surrounding environment. Its sufficient walking in a urban context close to buildings and car and the sensor already suffer their influence.
The accelerometer sensor measures the linear acceleration along the x,y and z axis represented in Figure 4.2 [24]. In general is adopted to capture the motion activities, as the orientation of display to put for example the device in landscape or vertical mode. The source of error derives mainly from the bias, the offset of its output signal from the true value.
The gyroscope calculates the angular velocities along the x,y and z axis as described in Figure 4.3. It useful to recover the correct orientation when the device is still in motion. In fact the estimate of the accelerometer is accurate only along stationary periods. While accelerometers measure linear acceleration as long as there is no rotation, gyroscopes generate an output signal directly proportional to the angular rate applied to the device.
In particular, the on board gyroscope measure the force generated by the Coriolis effect as shown in Figure 4.4. When an accelerometer is rotated, the projection of gravity acceleration is measured as well and there is no way to distinguish between the two different contributions. By using a gyroscope, angular rotations are measured while linear displacements are not. Accelerometers and Gyroscopes can be combined to create an Inertial Measurement Unit (IMU) able to reconstruct the movement.

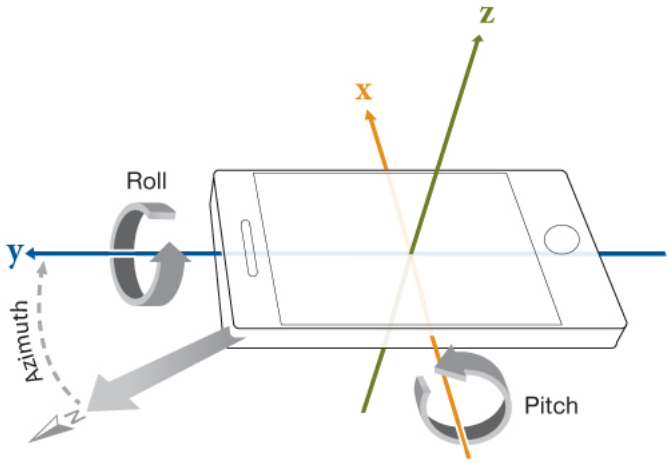

Figure 4.2: Smartphone rotational axis: the rotational angles among the x,y and z axis are respectively azimuth, pitch and roll.

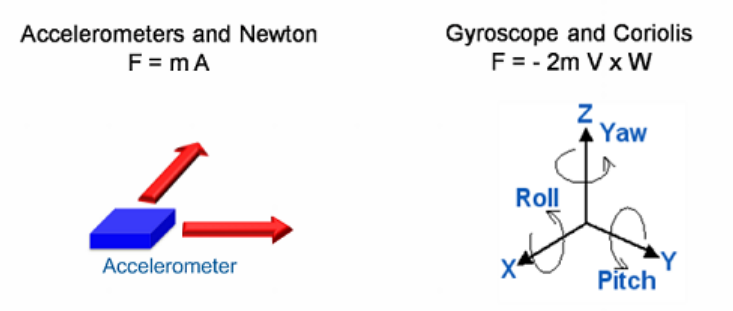

Figure 4.3: Accelerometer and Gyroscope functioning: V is the velocity of the particle with respect to the rotating system, and W is the angular velocity vector which has magnitude equal to the rotation rate ω and is directed along the axis of rotation of the rotating reference frame. [25]

The Gyroscopes use the Coriolis force to measure angular movement. When a mass moves in a particular direction with a velocity V and an external angular rate is applied (red arrow), the Coriolis effect generates a force, shown here with the yellow arrow, that causes a perpendicular displacement of the mass. The value of this displacement is directly related to the Angular rate applied.

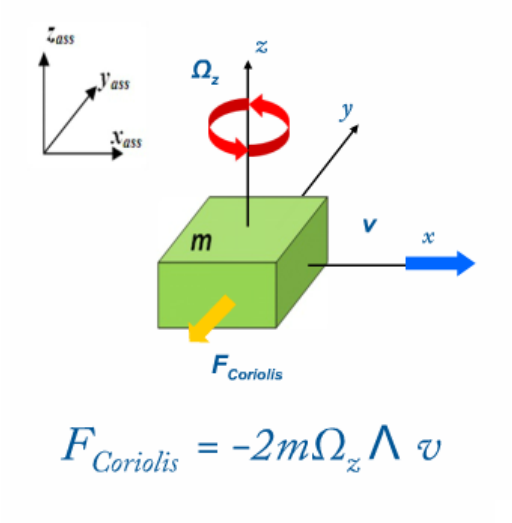

Figure 4.4: Coriolis Effect force in play: consider a mass moving in direction V. When an angular movement is applied (red arrow) the mass experiences a force in the direction of the yellow arrow as result of the Coriolis Effect. The light blue arrow represent the movement.[25]

The reliability problems of the gyroscope come from as the preceding sensor because of the magnetic interference and the bias and the numerical errors. In our specific field, the sum of all these errors lead two main problems: the initial estimate of the north and the bias over the time. Once received the information about the surrounding mountains the north direction is the crucial point to determine where the mountains are located. When this estimate contains evident error all the future work will be affected. During the classic use the user rotates and moves the device pointing towards the desired mountain to acquire the info related to. These movements are exactly the ones mentioned before where the sensors betray their vulnerabilities.
Regarding the localization, the services to recover the user position to improve the final experience are spread in almost every application. The scope can be the most various and each one can need several level of precision; starting from the estimate through the radio cell and Wifi connections, arriving to the most precise estimate exploiting also the GPS sensor helped from the previous two drastically decreasing the time to acquire the signal and then to recover the exact position. The final results are quite good as for typical navigation systems as for our specific scope. A recent localization system has been taken into account, The European Geostationary Navigation Overlay Service (EGNOS) [26]. EGNOS is an Europe venture into satellite navigation improves the open public service offered by the GPS. Known as a Satellite-Based Augmentation System, it provides both correction and integrity information about the GPS system, delivering opportunities for Europeans to use the more accurate positioning data for improving existing

services or developing a wide range of new services.

4.2 OpenCv LIbraries

Given all the advantages coming from a mature platform the interesting aspect leading us to the adoption of this particular API is the outcome performance obtained from natively implemented API in C. Thanks to this low level programming the control on the hardware is deeper and working on images the improvement is also more evident. Nothing is given for free, a little price has to be payed. This API has been released as a standalone application on the official android market Google Play Store. It's free and easy to install as all the other classical applications, and once installed and launched give only the possibility to check the current version installed and eventually to download and apply latest updates. Apart maybe other developer colleagues none obviously has already installed this application but the app is absolutely required to run the our. The solution is very simple and with low price effort.

4.3 The Activities

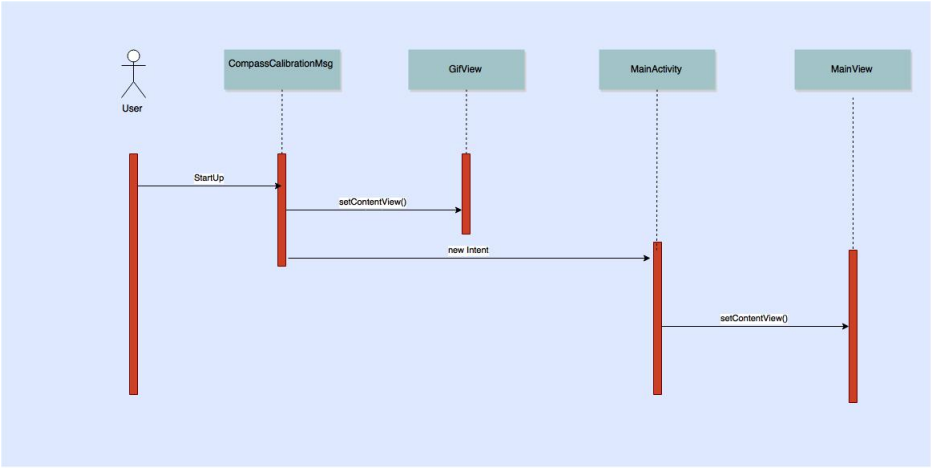

Figure 4.5: Sequence Diagram representing the execution flow of the Activities.

The application is based on two activities: the $CompassCalibrationMsg$ and the $MainActivity$ as shown in Figure 4.5. The purpose of $CompassCalibrationMsg$ is to suggest the Compass Calibration to the user as mentioned in Chapter 3. The corresponding View is the GifView, a simple View where the parameters of the GIF image are set, as the positioning, the duration, the refresh rate and the dimension. In the OnDraw method, the method responsible to custom drawing, the GIF is drawn on the screen as specified

by the previous parameters. As soon as in the $CompassCalibrationMsg$ the Compass calibration message has been shown a new Activity starts, the $MainActivity$. In the $MainActivity$ we find the most interesting implementation aspects about the GUI, the Sensors Managers, the Mountain models and Utilities containing the adopted algorithms. From now an overview of the $MainActivity$ implementation is exposed also through code snippets, focusing mostly on the matching algorithms running in the background asynchronous processes.

4.4 Sensor Managers

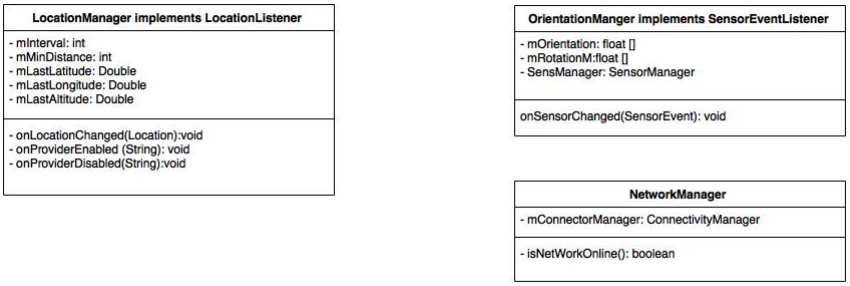

Figure 4.6: Class Diagrams of LocationManager, OrientationManager and NetworkManager.

The Sensor Managers are the LocationManager, the OrientationManager and the NetworkManager as shown in the Figure 4.6. They handle the embedded sensors presented in Chapter 3. LocationManager manages the GPS sensor checking the running status and recovering the position by means the altitude, longitude and latitude values. Every time new location is found a callback provides to update the coordinates in the $MainActivity$. The OrientationManager similarly to the previous one handles the axis orientation recovering the values of azimuth, pitch and roll and the cardinal directions. The NetworkManager provides control on the data connectivity, granting the availability to connect to our Polimi servers.

4.5 Camera Handlers

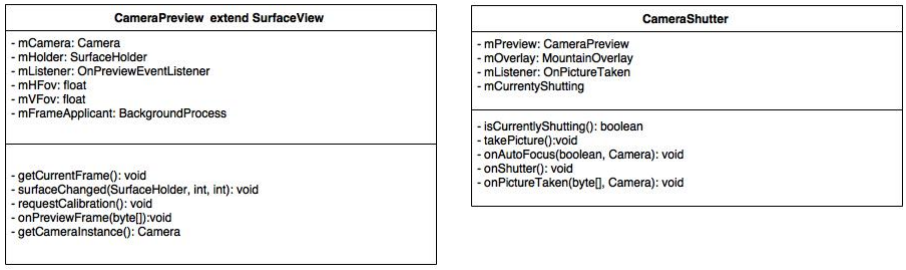

Figure 4.7: Class Diagrams of CameraPreview and CameraShutter.

As the *MainActivity* starts the CameraPreview is available. This class configures the preview visualized on the display. For instance provides a safe way to get an instance of the Camera object, it sets the optimal preview size and aspect ratio and it also rotates the preview when the device has been rotated as shown in Figure 4.7. Two important methods are implemented here: requestCalibration and onPreviewFrame. RequestCalibration is called by the *MainActivity* class when the execution flow needs the current frame and sets the PreviewCallback. When the preview frame is ready the onPreviewFrame is executed and it returns the required frame. At any time the user can take a photograph to store the current matching results. The CameraShutter class checks the status of the camera if it's currently shutting or not and performs the shutter setting the best configuration with respect to the available camera specifications.

4.6 The MountainOverlay View

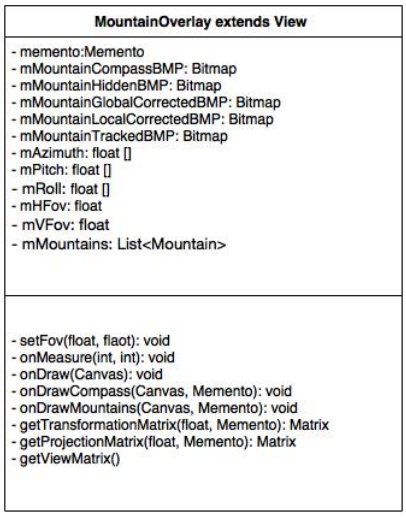

Figure 4.8: Class Diagrams of MountainOverlay.

The MountainOverlay is the View where the mountains are drawn on the preview camera frame. The most interesting methods are drawCompass and drawMountains, both called in the onDraw callback at every frame as described in Figure 4.8. DrawCompass draws the result of the magnetometer displaying the cardinal directions in the bottom of the screen. The drawMountains method instead exploits the mountain collection, set by the MountainDownloader, and the Memento to draws the peak markers on the preview frame according to the last available results of background matching processes. Memento is an inner class of MountainOverlay representing the orientation status at that precise moment. It contains the values of azimuth, pitch and roll together with the camera FOV. At each frame in drawMountain given the stored coordinates of the peaks in the Panorama 3D frame the Transformation Matrix based on the memento is calculated. Thanks to this matrix each vector peak *mountain* is projected from the Panorama 3D to the Camera 2D frame, the preview frame. The implementation of this coordinate system change is presented in the Listing 4.1:

```
1  float imgHFovRad = (float) Math.toRadians(imgHFov);

3  float imgWidthScaled = imgHFov*panoramaWidth/panoramaHFov;
   float imgSfX = imgWidthScaled/imgWidth;
5
   float imgHeightScaled = imgVFov*panoramaHeight/panoramaVFov;
7  float imgSfY = imgHeightScaled/imgHeight;
```

```
9
   Vector4 location = transformationMatrix.multiply(mountain);
11 final Vector3 peak = location.toVector3();

13 if (peak.z>0) {

15   float clipYMax = imgVFov/panoramaVFov;
     float clipXMax = (float) Math.cos(Math.PI/2 - imgHFovRad/2);
17
     float peakImgXScaledClip = peak.x/clipXMax;
19   float peakImgYScaledClip = peak.y/clipYMax;

21   float peakImgSXcaled = (peakImgXScaledClip+1)*imgWidthScaled/2;
     float peakImgYScaled = (1-peakImgYScaledClip)*imgHeightScaled/2;
23
     int x  = (int) (peakImgSXcaled/imgSfX);
25   int y = (int) (peakImgYScaled/imgSfY);

27   if ( x > 0 && y>0 && x < imgWidth && y < imgHeight){
       return new int[] {x,y};
29   }
   }
```

Listing 4.1: Transformation from Panorama 3D to Camera 2D Frame

where the panoramaHeight is 750 pixels, the panoramaWidth 3600 pixels, the vertical and horizontal panorama Fovs are respectively 75 degrees and 360 degrees and the imgHFOV is the vFov checked runtime and it depends on the device. All the mountain vectors containing the coordinates will be updated during the application execution by means the LocationUpdater processes.

4.7 The Mountain Downloader

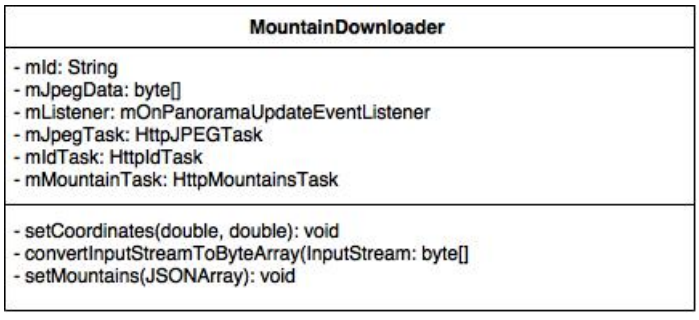

Figure 4.9: Class Diagrams of MountainDownloader.

When the GPS tag is available the instance of the MountainDownloader class, shown in Figure 4.9, manages the connection with the Polimi sever to

recover all the necessary information. Given the values of altitude, longitude and latitude recovered from the LocationManager the first HTTPRequest is executed passing the geographical position recovered. Once received the response containing the related panorama Id a second HTTPRequest to download the corresponding panorama JPEG image is performed. The last HTTPRequest is executed to download the JSON file containing the mountain details corresponding to the panorama Id received. In the method setMountains the peaks are bring from Panorama 2D to Panorama 3D and then the result is stored in a list of Mountain objects as shown in Listing 4.2

```
    float scaleFactor = (float) (Math.PI * 2f) / panoramaWidth;
32  float angleOrigin = (float) (-(scaleFactor*x) + Math.PI/2);
    float peaky = ((hHeight) - y)/(hHeight);
34  Vector3 peak = new Vector3((float)Math.cos(angleOrigin), peaky,
    (float)Math.sin(angleOrigin));
    mountains.add( new Mountain<Bitmap>(name, alt, distance, x, y, peak, id));
```

Listing 4.2: Transformation from Panorama 2D to Panorama 3D Frame

The scaleFactor is $degreePerPixel$ panorama density and the angleOrigin is the angle of the peak in the Operational Calculation System presented in the Chapter 3. The class Mountain represents the model with all the related info taken from the JSON file received.

4.8 Location Updaters

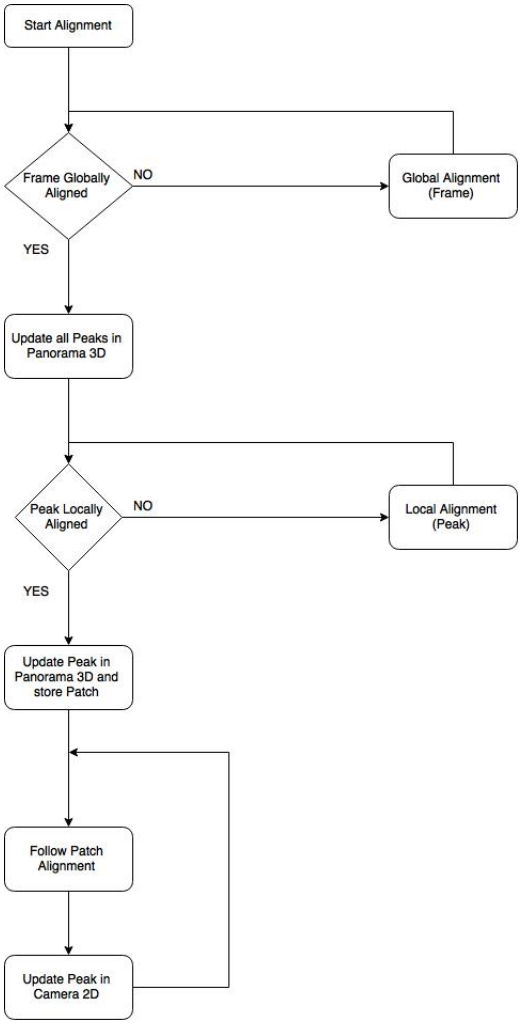

Figure 4.10: The Graph represents the alignment background processes based on LocationUpdaters.

The core implementation aspects are related to the matching algorithms inside the LocationUpdaters where the alignment based on the digital image processing occurs. Given the computational cost the algorithms runs in background threads by means the AsyncTask abstract class. The AsyncTask enables a proper and easy use of threads optimized directly from the OS. In the *MainActivity* there are three LocationUpdaters: the GlobalLocationUpdater, the LocalLocationUpdater and the FollowPatchLocationUpdater. They are activated in different steps during the application execution, some at the same time and multiple times and some other only once as shown

in the Figure 4.10.

4.8.1 Global Location Updater

The Global Location Updater is the first Location Updater based on image processing. As soon as all the information about mountains and panorama are available the current frame is required to the CameraPreview in startGlobalUpdate method. When the frame is received the onGlobalFrameReady method is executed. First the necessary parameters to perform the VccMatching are set, as the TransformationMatrix, the vertical and horizontal FOVs, the panorama received and the frame just required. Then the background thread GlobalLocationUpdater starts. The are several steps to pass through this GlobalLocationUpdater and now will be presented the most interesting ones. To find the correct overlapping between frame and panorama the first must be scaled to obtain the same *degreeperpixel* density of the latter as shown in Listing 4.3.

```
36  final float pPixelByDegree = panoramaWidth / 360f;
    final float fPixelByDegree = originalFrameWidth / hFov;
38  final float scaleFactor = pPixelByDegree / fPixelByDegree;

40  it.polimi.snowwatch.utils.Bitmap<android.graphics.Bitmap> scaledPicture =
            new BitmapWrapper(android.graphics.Bitmap.createScaledBitmap(
42          cameraPicture.getWrappedBitmap(),
            Math.round(cameraPicture.getWidth() * scaleFactor),
44          Math.round(cameraPicture.getHeight() * scaleFactor), false));
```

Listing 4.3: Scaling frame to match the panorama $degreeperpixel$ density.

During our test cases the originalFrame dimension is 1920x1080 pixels and the vertical and horizontal FOVs of the device are 53 degrees and 61 degrees. To extract the corresponding panorama patch, the frame center is projected on the panorama as presented in Listing 4.4.

```
    Matrix invTransMatrix = transformationMatrix.invert();
46  Vector4 peak4 = invTransMatrix.multiply(new Vector3(0, 0, 1));
    Vector3 peak = peak4.toVector3();
48
    int[] panCoordinate =
    ImageToolkit.fromPanoramaFrame3dToPanoramaFrame2d(peak);
50  int panx = panCoordinate[0];
    int pany = panCoordinate[1];
```

Listing 4.4: Find the frame center projected on the panorama.

The invTransMatrix is the inverse of the TransformationMatrix obtained from the Memento in the MountainOverlay instance where the orientation estimate of the processed frame is available. Thanks to the invTransMatrix the peak coordinates pass from Camera 3D to Panorama 3D Frame. The implementation to pass from Panorama 3D to Panorama 2D Frame is shown in the Listing 4.5.

```
52  double px = (float) peak.x;
    double py = (float) peak.z;
54
    float angleOrigin = (float) Math.acos(px);
56
    if (py<0)
58    angleOrigin = - angleOrigin;

60  float angleOriginDeg = (float) Math.toDegrees(angleOrigin);
    if(angleOriginDeg < 0)
62    angleOriginDeg = 360 + angleOriginDeg;

64  float angle = -angleOriginDeg + 90;
    if (angle<0) angle = 360 + angle;
66
    int pixelPerDegree = panoramaWidth/panoramaHFov;
68
    int panx = (int) (angle * pixelPerDegree);
70  int pany = (int) (renderHeight/2*(1-(float) peak.y));
```

Listing 4.5: Transformation from Panorama 3D to Panorama 2D Frame.

The Gaussian Kernel function implemented as described in Listing 4.6 is set to fit the extracted patch dimension:

```
    final int mu = Math.round(size/2);
72  float [][] shiftedkernel = new float[size][size];

74  for ( int c=0; c<size; c++){
      for (int r=0; r<size; r++){
76      shiftedkernel[c][r] = (float) Math.exp(-kernelValue*(-Math.pow(c-mu,
        2)-Math.pow(r-mu, 2)));
      }
78  }
    gaussianKernel = mFactory.shiftMatrix(shiftedkernel, 0, 0, size, size, true, true,
    false);
```

Listing 4.6: The GaussianKernel implementation.

In fact the kernel size is equal to the scaledPicure size, in the test cases is 630x350 pixels and the kernelValue is $5 \cdot 10^{-5}$.
Finally the VectorCrossCorrelation, all the code is shown in the Listing 4.7 algorithm is executed.

```
80    float [][] fPanorama = factory.imageToMatrix(panorama, panx, pany,
      panoramaPatchSize, panoramaPatchSize, true, false, false);

82    float [][] fexPanorama = ImageToolkit.createMatrix(panoramaPatchSize,
      panoramaPatchSize);
      float [][] feyPanorama = ImageToolkit.createMatrix(panoramaPatchSize,
      panoramaPatchSize);
84    ImageToolkit.Convolve(fPanorama, ImageToolkit.HorizontalEdges, fexPanorama, 5);
      ImageToolkit.Convolve(fPanorama, ImageToolkit.VerticalEdges, feyPanorama, 5);
```

```
86      fPanorama = null;

88      fexPanorama = ImageToolkit.SkylineRoman(fexPanorama, panSkylineTh);
        feyPanorama = ImageToolkit.SkylineRoman(feyPanorama, panSkylineTh);
90
        int scaledPictureWidth = scaledPicture.getWidth();
92      int scaledPictureHeight = scaledPicture.getHeight();
        int fx = Math.round(scaledPictureWidth/2);
94      int fy = Math.round(scaledPictureHeight/2);

96      float [][] fPicture = factory.imageToMatrix(scaledPicture, fx, fy,
        scaledPictureWidth, scaledPictureHeight, false, false, false);
98
        float [][] fexpicture = ImageToolkit.createMatrix(scaledPictureHeight,
        scaledPictureWidth);
100     float [][] feypicture = ImageToolkit.createMatrix(scaledPictureHeight,
        scaledPictureWidth);
        ImageToolkit.Convolve(fPicture, ImageToolkit.HorizontalEdges, fexpicture, 5);
102     ImageToolkit.Convolve(fPicture, ImageToolkit.VerticalEdges, feypicture, 5);
        fPicture = null;
104
        //Edge Filtering
106     fexpicture = ImageToolkit.Threshold(fexpicture, picTh);
        feypicture = ImageToolkit.Threshold(feypicture, picTh);
108
        //Skyline Detection
110     fexpicture = ImageToolkit.Skyline(fexpicture, picSkylineTh);
        feypicture = ImageToolkit.Skyline(feypicture, picSkylineTh);
112
        float potentialEdge = ImageToolkit.potentialEdge(fexPanorama, feyPanorama);
114
        float [][] fexpictureResized = resizeImage(fexpicture, panoramaPatchSize);
116     float [][] feypictureResized = resizeImage(feypicture, panoramaPatchSize);

118     //Square
        ImageToolkit.Square(fexPanorama, feyPanorama);
120     ImageToolkit.Square(fexpictureResized, feypictureResized);

122     //DFFT
        Fft2D.transform(fexPanorama, feyPanorama);
124     Fft2D.transform(fexpictureResized, feypictureResized);

126     //Conjugate
        ImageToolkit.Conjugate(fexPanorama, feyPanorama);
128
        //Product
130     ImageToolkit.Multiply(fexpictureResized, feypictureResized, fexPanorama,
        feyPanorama);
        fexPanorama = null;
132     feyPanorama = null;

134     //IDFFT
        Fft2D.inverseTransform(fexpictureResized, feypictureResized);
136
```

```
        //Real part needed
138     feypictureResized = null;

140     //Gaussian Kernel filter
        ImageToolkit.gaussianKernel(fexpictureResized, shiftedKernel);
142
        //Max
144     ImageToolkit.ImagePoint[] points = ImageToolkit.getMaximum(fexpictureResized,
        1);

146     if ( points[0].value<potentialEdge) return null;

148     int xoffset = (int) (fexpictureResized.length - points[0].x);
        if (xoffset >= fexpictureResized.length/2) xoffset -= fexpictureResized.length;
150     int yoffset = (int) (fexpictureResized[0].length - points[0].y);
        if (yoffset >= fexpictureResized[0].length/2) yoffset -=
        fexpictureResized[0].length;
```

Listing 4.7: The VectorCrossCorrelation implementation.

The threshold parameters $picTh$ and $picSkylineTh$ adopted are respectively 0.15 and 0.28.

At the end there is the check to understand if the viewing mountains are really visible or there are some obstacles occluding the peaks. If the max extracted from the resulting matrix representing the VectorCrossCorrelation output is higher than the potentialEdge then the quality of alignment is enough consistent and all the peaks are updated. With the obtained offsets the peak in the Panorama 2D frame are updated and as in the beginning stage the peaks are projected from the Panorama 2D into the Panorama 3D Frame. From now on in the MountainOverlay View the peaks will be displayed with the fixed position just updated.

4.8.2 Local LocationUpdater

The GlobalLocationUpdater is executed only one time to obtain the global alignment. The LocalLocationUpdater instead is performed multiple times, once per each peak. The execution flow is similar: the steps are the same but the source images are not the entire frame and the panorama patch corresponding to the overlapping of the entire frame. The source images of the matching process are the patches containing only the single peaks. At the end after the resulting offset are applied and the peaks are updated in Panorama 3D frame per each peak a patch from the current frame is stored. These patches will be necessary in the future Following Patch step. The patches are 50x50 pixels and are store inside the corresponding Mountain object.

4.8.3 FollowPatch LocationUpdater

Until all the peaks have been processed in the LocalLocationUpdater the background threads still continue. For each mountain already locally fixed the last FollowPatchLocationUpdater starts. Here the OpenCV libraries are exploited: the input are the extracted patch from the frame on which we performed the local alignment and the current frame. The implementation is shown in the Listing 4.8.

```
152   Mat img = fromBitmapToMat(image);
        Mat templ = fromBitmapToMat(m.getPatch());
154
      int resultCols = img.cols() - templ.cols() + 1;
156     int resultRows = img.rows() - templ.rows() + 1;
        Mat result = new Mat(resultRows, resultCols, CvType.CV_32FC1);
158
        Imgproc.matchTemplate(img, templ, result, matchMethod);
160
        Core.normalize(result, result, 0, 1, Core.NORM_MINMAX, -1, new Mat());
162     MinMaxLocResult mmr = Core.minMaxLoc(result);

164     Point matchLoc = mmr.maxLoc;

166   double xOff = matchLoc.x + templ.cols();
        double yOff = matchLoc.y + templ.rows();
168
        int px = (int) xOff - templ.cols()/2;
170     int py = (int) yOff - templ.rows()/2;
```

Listing 4.8: The FollowingPatch implementation.

The Imgproc.matchTemplate is core method where takes place the matching: the best overlapping of the patch inside of the current frame. In Core.normalize occurs the normalization and MinMaxLocResult.MatchLoc gets the maximum from the resulting matrix of the matching. The $matchMethod$ adopted is $CvTmCoeffNormed$. To maintain the experience as smooth as possible both input images are scaled with a scale factor of 0.25.

Chapter 5

Experimental Study

In this chapter a sample collection is analyzed to evaluate the performances of the developed application. The precision of the estimate is computed varying the operating parameters and different thresholds are set to classify the resulting peaks. Now I will present the test approach with the related difficulties and all the features needed to build the Test Environment. At the end some consideration about results will be exposed.
To analyze the precision and robustness of MountainWatch an ad-hoc testing environment has been built. Performing test on mobile platforms brings additional difficulties with respect to the "typical" applications. In particular the application is based on data recovered from sensors and cameras carrying additional labor also for the simple debugging phase where a virtual running instance in the developing environment is not possible. The peak detection evaluation directly on the device is not possible for many reasons. The correctness of the peak positions on the Camera 2D Frame can be evaluate only thanks to a groundtruth. The groundtruth is the set of all the samples where the alignment among the Panorama 2D and the Frame 2D is performed manually, so it represents the perfect alignment that application can reach. This manual alignment must be created manually for the entire Data Set to be analyzed and it is performed with an application presented in this Chapter in the section Manual Alignment.

5.1 Data sets

The Data Set is composed by photographs taken from the bank of the Como city with a typical landscape of mountains overlooking the lake and near a skiing area in Valtellina with peaks snow covered captured from high altitudes. The photographs are 100 with various weather conditions, from none or minimal to massive cloud presence, from clean shutter with no obstacle to picture with cumbersome object occluding the view as shown in

the Figure 5.1.

Figure 5.1: Samples taken from the Dataset.

5.1.1 Collecting Samples

The GPS coordinates are necessary at the boot of the application, and eventually at each relevant position change, to download the Panorama 2D and the contained peaks in that specific location. The sensor readings about the orientation are needed during all the application runtime. In the Test Environment must be reproduced the identical situation as the application runs exactly on the device in the same position, orientation and capturing the same frame. The idea is to capture the Memento, as presented in the Chapter 4, to store the status of the sensors at each taken frame. To recover the test samples a series of frames is stored with an interval of 1 second among them. The hypothesis is that the GPS coordinates does not change so during the entire sequence the GPS tag is unique. For each frame the Memento is stored keeping track of azimuth, pitch and roll together with the GPS tag relative to the entire sequence. These details are stored inside the local storage of the device in a JSON file together with the frames.

To store the samples a new feature to the original application is added. It is accessible from the main view simply via a Button as the other functionalities. Before to start the GPS tag must be available but it is not necessary to have already downloaded the Panorama 2D. Through an AsyncTask, used for the other LocationUpdater as presented in the Chapter 4, at each second the screenshot of the visible camera preview is stored, without the MountainOverlay layer. The result of LocationUpdater printed by the MountainOverlay is not needed because the same results will be reproduced "offline" in the Simulator, the reproduction of the mobile application in the Test

Environment. This layer also with peak markers and compass directions added artificially are misleading elements for the matching test. Given the device adopted for the test, OnePlus One, each frame of the Screenvideo has Full HD resolution, 1920x1080 pixels. The shots are available in the folder *emulated/MountainWatch/SampleTest*.

5.1.2 Manual Alignment

To create the groundtruth a tool has been developed by a collaboration of several colleagues and Homeria [27]. It's a web application able to align manually the Panorama 2D and the Camera 2D Frame, so the sample frames stored for the test in the application and the corresponding panorama. Through a simple graphical interface is possible to align the two pictures and it returns the results in a JSON file available for the download. The first operation is to upload the sequence of frames and the relative JSON file from the device. Then the Camera 2D is set in the center of the page in the foreground and the Panorama 2D is put in the background. With a simple mouse scroll movement the user can align the frame over the panorama as shown in Figure 5.2.

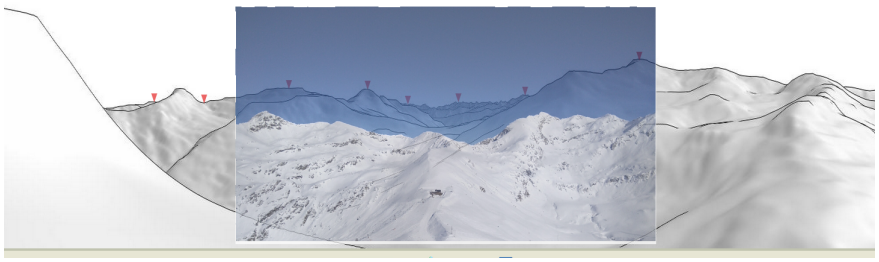

Figure 5.2: The alignment of the frame with the panorama.

Once the best overlapping alignment is found also the position of the Panorama 2D is fixed. The DEM as presented in the Chapter 3 does not correspond perfectly to the real mountain profiles. This application provides a functionality to stretch the Panorama to be correctly aligned with the Camera frame. To obtain the pixel position of all the visible peaks in the Camera 2D Frame the estimate of the frame's center on the panorama has to be found when the Panorama Stretching, described in Figure 5.3, is performed.

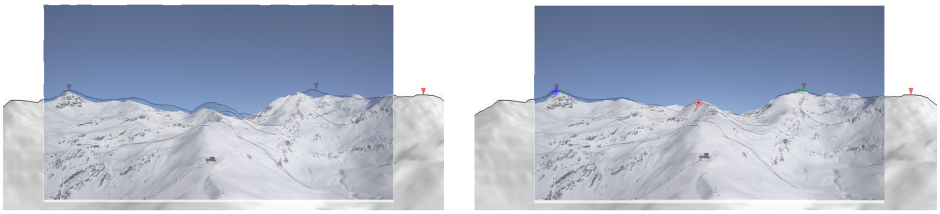

Figure 5.3: The frame is stretched to overlap perfectly the panorama.

At the end the JSON file containing the pixel coordinates of all visible peaks is available. The stretching leads to calculate the resulting center of the frame. An estimate of the least square error is recovered.

5.2 Metrics

The matching process is based on several steps so the evaluation must takes care of the results at each stage as well as the final result. The evaluation of embedded sensor consists of the calculation of the distances among the resulting peaks coming from the readings sensor in the Camera 2D with respect to the resulting peaks obtained from the Manual Alignment. The parameters are the Euclidean Pixel Mean Distance (EPMD), the Degree Mean Distance (DMD), the Center Euclidean Pixel Mean Distance (CEPMD) and the Center Degree Mean Distance(CDMD). The EPMD is the mean in pixels of the Distance Error peak by peak of each photograph; the DMD is the same Distance Error in degrees. The CDMD is the Degree Error among the picture center projected on the Panorama 2D with respect to the picture center calculated with the Manual Alignment representing the best approximation of the center after the image Stretching. The CEPMD is the same Error in pixels. The EPMD and DMD estimate the absolute error visible from the user on the picture. These parameters contain an inside intrinsic error coming from the downloaded panorama: as described in the previous chapters the DEM and its projection have some imprecision and the algorithm cannot reach the absolute precision with these initial data. To better evaluate the goodness of the algorithm both will be take into account. The Azimuth Error and Pitch Error are respectively the error along the horizontal and vertical axis. After each stage of alignment process the Confusion Matrix (CF), described in the Table 5.4, is calculated.
In this model each peak of all series pictures is evaluated: the True Positive(TP) represents the peak marked thanks to the sensors matching and matched also in the groundtruth, without taking care of the Distance Er-

	Relevant	Non-relevant
Retrieved	true positive (TP)	false positive (FP)
Not retrieved	false negative (FN)	true negative (TN)

$$P = \frac{TP}{retrieved} = \frac{TP}{TP + FP}$$

$$R = \frac{TP}{relevant} = \frac{TP}{TP + FN}$$

Figure 5.4: The Confusion Matrix allows visualization of the performance of the algorithm: P and R are respectively the Precision and the Recall.

ror but only the presence. The False Positive (FP) is the peak found from the Matching and not from the groundtruth. The True Negative(TN) is the peak not found from both sensor matching and groundtruth. The False Negative (FN) is the peak retrieved in the groundtruth set and not in the sensor matching. The values of Precision and Recall give an overview of the result goodness.

5.2.1 Test Flow Execution

For each sequence thanks to the collected information obtained as explained before in Section Collecting Samples the Panorama 2D is downloaded with the details about the contained peaks. For each photograph of the sequence the same execution is reproduced exactly as on the mobile device. At first the estimation of the Sensor Matching is reproduced thanks to the Transformation Matrix obtained from the Memento. Then the Global Updater and Local Updater are proposed again exactly as in the mobile application. The evaluation of the last phase of Following Patch is a different from the original execution on the device. During the execution of the application after the Local Location Updater has finished all the corrected mountains are tracked in the following frames. Given the entire sequence of 10 frames taken in a short range of time the extracted patch from previous steps can be tracked on the entire sequence, not only the frame successive frames. For

each extracted patch we perform the tracking on all the other nine frame augmenting the test cases. For example when the fifth frame is analyzed we will search its peak patches not only in the subsequent frames 6,7,8,9 and 10 but also in the previous frames 1,2,3 and 4. Once all these estimations are calculated a threshold must be set to declare if the resulting peak is correctly matched or not and how much is precised. As described in Table 5.1 there are 3 threshold levels to classify the Degree Error ϵ of resulting peak detection.

Level of Precision	**Degree Error**
High	$\epsilon < 3$
Medium	$3 < \epsilon < 5$
Low	$5 < \epsilon < 7$
Not Found	$\epsilon > 7$

Table 5.1: Degree Error Classification.

5.2.2 Operating parameters

The Test is performed evaluating the algorithm presented in the Chapter 3 changing the operating parameters as presented in the Table 5.2. The values in bold defines the default value adopted for the final release and that gives the best evaluation results.

Full name	**Parameter**	**Tested values**
Panorama Edge Filtering Threshold	ρ_p	**0.1**, 0.2
Picture Edge Filtering Threshold	ρ_r	0.2, **0.3**
Panorama Skyline Filtering Threshold	b_p	0.2, **0.3**
Picture skyline Filtering Threshold	b_r	0.3, **0.4**

Table 5.2: Operating parameters (defaults in bold).

All the operating parameters have been tested. The default configuration is set and one parameter at time is evaluated. For each parameters all values are tested estimating the DMD: the default value corresponds to the minimum DMD found. Table 5.3 represents the other invariant parameters.

5.3 Results

The Dataset is divided in 10 sequences are analyzed taking into account some important aspects as the landscape context, the obstacles present in the scene, the presence of clouds on the skyline horizon and the precision of the initial Panorama 2D how much is similar and accurate the DEM with respect to the real mountains profile first and contour after. As shown in

Full name	Adopter values
Render global patch size	750
Render local patch size	250
Picture local patch size	100
Patch Scale Factor	0.25
Extraction Patch Size	50

Table 5.3: Invariant operating parameters .

Table 5.4 and 5.5 for each sequence the estimates of DMD, PMED, CDMD and CPMED explained in the Section Metrics together with the value of the Precision and Recall. The first table refers to the peak estimation after the Sensor Matching to evaluate the precision of sensors; the latter represent the results after the Local VCC Matching. The entire collection refers to the same location revealing the fluctuating behavior of the entire detection process. The situation differs a lot one sequence from the others about the context, the weather conditions and presence of obstacle in the FOV. Also inside the same sequence the results can be really different: the DEM accuracy and the compass measures changing over the time also during a small periods of time to perform shutter sequence, about a dozens of seconds.

Sequence	DMD	PMED	CDMD	CPMED	Recall	Precision
One	15.85	490	13.70	428	0.57	0.13
Two	10.4	325	11.65	364	0.81	0.77
Three	15.85	490	13.70	428	0.86	0.94
Four	3.75	114	4.40	134	0.89	0.89
Five	3.37	102	3.61	110	0.91	0.88
Six	5.03	153	4.81	146	0.59	0.96
Seven	4.29	130	4.45	135	0.77	1
Eight	5.91	180	6.13	186	0.60	0.92
Nine	12.63	684	12.34	428	0.79	0.91
Ten	4.40	122	4.69	142	0.85	1
Mean	**7.09**	**246**	**7.08**	**222**	**0.76**	**0.84**
Median	**5.18**	**157**	**4.9**	**1449**	**0.82**	**0.91**

Table 5.4: Result obtained in the Sequences after the Sensor Matching phase.

In sequences One, Two, Four, Five, Six, Eight, Nine and Ten the improvement thanks to VCC Matching is positive and evident. In certain cases the initial error due to the compass estimate is so pronounced that the improvement is not sufficient to reach the optimal peak detection as in sequences One and Two. In the sequence One the presence of clouds is low, none obstacles and good wide view but the Panorama 2D is not accurate. In sequence Two the Panorama 2D is quite accurate but the there are several obstacles

and restricted narrow view. In the sequences Four, Five and Ten the clean and wide view, the absence of obstacles and clear skyline together with the accurate DEM lead to good results. In sequence Three, Seven and Nine the improvements obtained from the image processing are negative but analyzing deeply the single frame the real final detection precision is not always worse than the one obtained through the sensors.

Seq	**DMD**	**PMED**	**CDMD**	**CPMED**	**Recall**	**Precision**	**Imp**
One	7.45	332	7.25	226	0.75	0.70	45%
Two	5.27	164	5.47	170	0.81	0.77	53%
Three	5.76	175	6.31	192	0.81	0.77	10%
Four	2.56	72	1.57	47	0.93	0.96	64%
Five	2.46	75	1.97	60	0.91	0.94	45%
Six	3.07	93	2.61	79	0.53	0.96	46%
Seven	7.61	231	7.19	219	0.72	0.95	10%
Eight	3.12	97	3.31	103	0.75	0.94	47%
Nine	1.93	58	1.37	41	0.84	0.91	89%
Ten	4.68	142	2.73	83	0.92	1	42%
Ma	**4.39**	**143**	**3.97**	**122**	**0.78**	**0.89**	**45%**
Me	**3.9**	**119**	**3.02**	**93**	**0.81**	**0.94**	**45%**

Table 5.5: Result obtained in the Sequences after the VCC Matching phases.

In sequence Three the result of the VCC Matching is not representative: looking at the single frames the frame 1,2,3,4 and 8 the final DMD is under the 3 degrees. The remaining frames due to the DEM imperfection the alignment leads to the negative improvement. In sequence Seven the results of the entire series are negatives due to the presence of obstacles. The initial error compass estimate is quite high and in the frames 7, 8 and 9 with a good skyline profile and without obstacle the improvement lead to DMD less then 3. In the other frames the large imprecision given from the compass affects the image processing resulting not able to recover the correct alignment getting worse detection.
Now we consider the Data Set not any more divided by sequence but we analyze the entire set of peaks. As shown in the Figure 5.5 the total amount of True Negative peaks available is very high. This result leads to the consideration of the large quantity of peaks that are present in that specific geographic area but not directly visible from the viewer. This estimation is the first evident difference with respect the actual applications available in the Google Playstore where all the possible mountains are detected. In Figure 5.6 is shown the final result where all the peaks are classified as described in Table **??**. The classical arithmetic mean tends to not describe faithfully the goodness of our system and tends to under estimate the results. In Figure 5.7 can be appreciated the general improvement starting from the

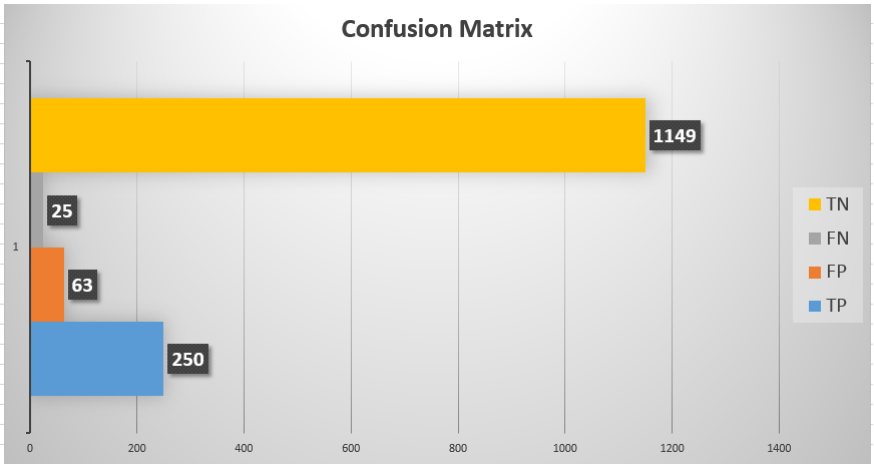

Figure 5.5: The Confusion Matrix representing the entire Dataset.

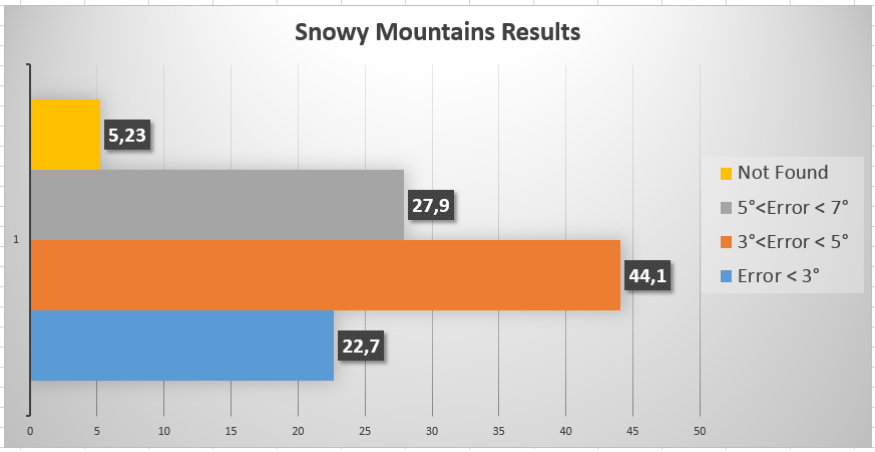

Figure 5.6: The classification of all the resulting peaks divided in four categories.

Sensor Matching arriving to Following Patch stages.
The stable trend of the change from the last two steps is not related to a neutral detection improvement but to the adopted approach. In the last phase the purpose of this different strategy is to reduce the time of execution and not anymore to augment precision. The target is to maintain the same detection of the Local VCC matching granting a better smoothness. The frame rate during the Following Patch step decreases but still remains acceptable granting the user experience satisfactory.
For each sequence of the Dataset the execution time is analyzed. The Sensor matching consists of a not computationally heavy calculations in fact there is no latency and it's immediately displayed. The stages taken into account time are the Global VCC Matching, Local VCC Matching and the Following Patch. During the data collection at each of these stages the duration of

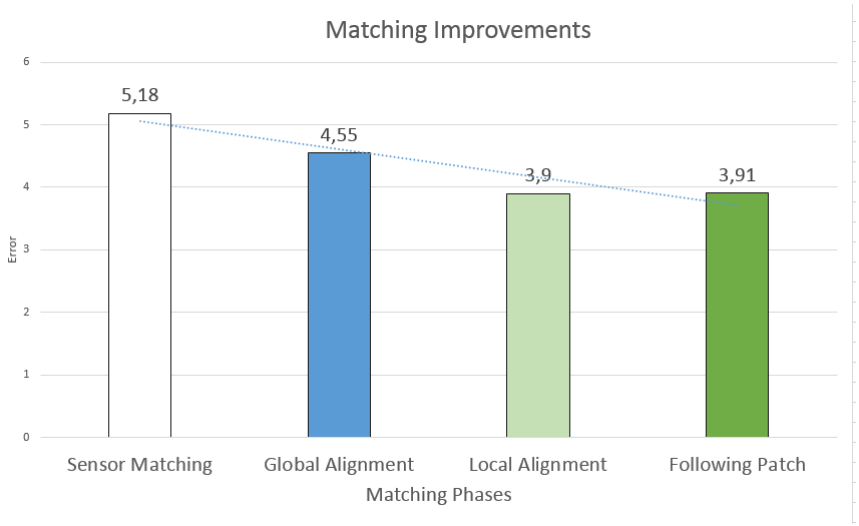

Figure 5.7: The Bar Chart represents the resulting DMD Error of the entire Data Set.

the alignment of all the peaks present in each frame is stored. In the Global Matching the time is fixed: we align the Panorama 2D and the Camera 2D one single time. In the other stages the duration refers to the alignment of all the peaks visible in that specific frame so it can vary according to the quantity of visible mountains. A weighted estimation is calculated simply dividing the amount of time by the number of detected peaks. In Figure 5.8 are represented the overall mean execution times of the entire Data Set of the 3 phases.
Though Local VCC Matching obtains similar errors, the alignment of a single peak with Local VCC Matching requires $1.71s$ while a peak alignment with Following Patch approach requires $0.81s$, i.e. an 52% time reduction.

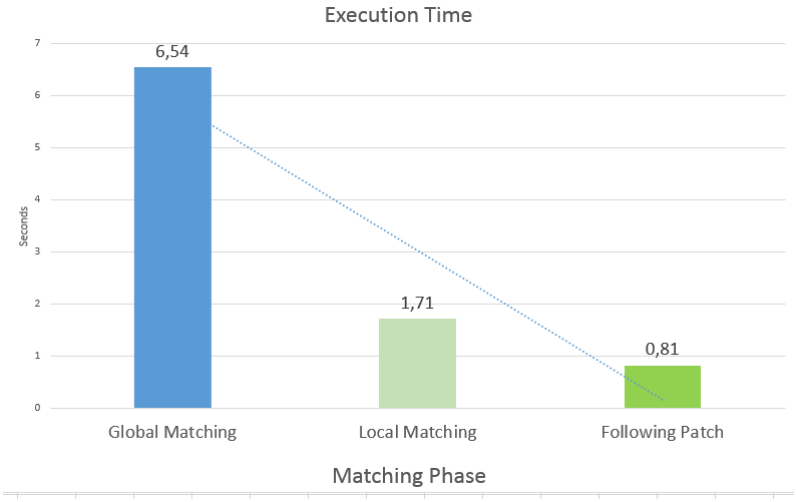

Figure 5.8: The Bar Chart represents the resulting DMD Error of the entire Data Set.

Chapter 6

Conclusions and Future Work

In this work a AR mobile system for the estimation and the detection of peaks based on sensors first and image processing after is presented. The system as shown in the Chapter 5 detects the peaks with a $precision = 0.80$ and $recall = 0.90$. It reaches a Degree Error $\epsilon < 3$ in 23% and $3 < \epsilon < 5$ in 44% of the data set. The precision of the final detection depends from many factors and the most important are the mounted sensors, the camera and the computing power. The user experience results to be really comfortable using recent devices granting a general level of smoothness. Thanks to the available power computing and the great precision of the mounted sensors the subsequent image processing can overcome the initial starting errors about orientation and localization. The direction of the components used for mobile device lead us an happy future given the huge improvement in the last few years giving the possibility to improve the reached results adopting new more precise matching algorithms and greater general smoothness.

6.1 Future Enhancements

The main aspects of this work are related to the precision reached in the final mountain tags and the possibility to exploit passively as much as possible these results. The possibility for the users to directly share the photographs containing the tagged mountains on the main social platforms inside our application is already arranged. It can be an involvement for people to share easily and immediately the resulting mountain detection. In this way releasing this additional feature the user experience is richer and at the same time the data acquisition results simplified.
Now some considerations about new different technical strategies are presented.

6.1.1 Matching Performances

A possible additional feature to improve the matching algorithm performance is also to give the possibility to the user the correct manually the final alignment. With an intuitive interface the user can drag the resulting marker more precisely to the correct peak position only over the peak profile exploiting for example an audio signal guide and visual hints. Once acquired the new position it will substitute the previous one in the Camera 3D Frame correcting the peak label. In this approach the role of the user changes. He becomes an important actor in the alignment process helping our system in a typical touch task for computer system.

A specific work on the sensor precision evaluating the possibilities is presented in [23] where a Drift and Noise Removal Filter (DNRF) is described. It is implemented by sensor fusion of gyroscope, magnetometer and accelerometer which minimizes the drift and noise in output orientation. A numerical error correction approach is also mentioned to minimize the errors caused by gyro signal integration. The orientation result obtained by proposed DNRF method are smooth and less noisy as compared to digital compass. The adoption of this system can reduce the initial errors coming from the sensors.

6.1.2 Local Digital Elevation Model

The Panorama 2D derived from the DEM is a crucial aspect of future improvements about precision and execution time. A possibility to reduce the global execution time is that the Panorama 2D can be downloaded with the mountain profiles already extracted without processing the image locally. Performing these operations of edge extraction and edge filtering server side is possible to adopt heavier but more accurate algorithms also obtaining better performance during the alignment phase. Another expensive solution is to develop a owner panorama generator inside the application and available locally. In this way the precision of the peak detection can be augmented reaching better alignments and enhancing the correspondence between the 3D model and real mountains environment. A local database installed on the device is concretely feasible given the storage capacity of new generation devices. The user has the possibility to select the preferred regions according to his geographical position and download only the DEM of the interesting zone just once. The DEM locally stored allows to not need to download the Panorama 2D every time reducing the execution time of the final peak tagging. A possible solution can be to generate only the projection from the 3D model, the Panorama 2D, at execution time without the need of data connectivity. This solution has been already adopted with good results in several other similar applications giving more freedom to the user.

6.1.3 Safeguard and Environmental Modeling

An idea can be the creation of challenge contests suggesting people to explore and take shutters of a certain mountainous areas on which some environmental associations are strictly interested. The motivations can be simply the lack of information of that particular zone or maybe a more recent data to perform cyclic long-term analysis. An example can be to put up for grab as rewards a quantity of premium points thanks to which the user can access to discounts from the same associations mentioned before (for example the Club Alpino Italiano CAI) for event participation or gadgets.

Bibliography

[1] Roman Fedorov, Piero Fraternali, and Marco Tagliasacchi. Snow phenomena modeling through online public media. *Image Processing, 2014 IEEE International Conference on*, page 2179.

[2] A. M. Turing. Computing machinery and intelligence. *Mind*, 59(236):pp. 433–460, 1950.

[3] Alexander J. Quinn and Benjamin B. Bederson. Human computation: a survey and taxonomy of a growing field. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1403–1412, New York, NY, USA, 2011. ACM.

[4] P. Fraternali, A. Castelletti, R. Soncini-Sessa, C. Vaca Ruiz, and A. E. Rizzoli. Putting humans in the loop: Social computing for water resources management. *Environ. Model. Softw.*, 37:68–77, November 2012.

[5] Edith Law and Luis von Ahn. Input-agreement: A new mechanism for collecting data using human computation games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pages 1197–1206, New York, NY, USA, 2009. ACM.

[6] Lorenzo Porzi, Samuel Rota Buló, Paolo Valigi, Oswald Lanz, and Elisa Ricci. Learning contours for automatic annotations of mountains pictures on a smartphone. In *Proceedings of the International Conference on Distributed Smart Cameras*, ICDSC '14, pages 13:1–13:6, New York, NY, USA, 2014. ACM.

[7] James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[8] Srikumar Ramalingam, Sofien Bouaziz, Peter Sturm, and Matthew Brand. Geolocalization using skylines from omni-images. In *Proceedings of the IEEE Workshop on Search in 3D and Video, Kyoto, Japan*, 2009.

[9] Roman Fedorov, Piero Fraternali, and Marco Tagliasacchi. Mountain peak identification in visual content based on coarse digital elevation models. In *Proceedings of the 3rd ACM International Workshop on Multimedia Analysis for Ecological Data*, pages 7–11. ACM, 2014.

[10] Lionel Baboud, Martin Cadik, Elmar Eisemann, and Hans-Peter Seidel. Automatic photo-to-terrain alignment for the annotation of mountain pictures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), oral presentation*, 2011.

[11] Prospero C. Naval Jr, Masayuki Mukunoki, Michihiko Minoh, and Katsuo Ikeda. Estimating camera position and orientation from geographical map and mountain image. In *38th Research Meeting of the Pattern Sensing Group, Society of Instrument and Control Engineers*, pages 9–16, 1997.

[12] Georges Baatz, Olivier Saurer, Kevin Köser, and Marc Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Proceedings of the 12th European conference on Computer Vision - Volume Part II*, ECCV'12, pages 517–530, Berlin, Heidelberg, 2012. Springer-Verlag.

[13] I. Celino, D. Cerizza, S. Contessa, M. Corubolo, D. Dell'Aglio, E.D. Valle, and S. Fumeo. Urbanopoly – a social and location-based game with a purpose to crowdsource your urban data. In *Privacy, Security, Risk and Trust (PASSAT), 2012 International Conference on and 2012 International Confernece on Social Computing (SocialCom)*, pages 910–913, Sept 2012.

[14] Yetao Huang, Yue Liu, and Yongtian Wang. Ar-view: An augmented reality device for digital reconstruction of yuangmingyuan. In *Mixed and Augmented Reality - Arts, Media and Humanities, 2009. ISMAR-AMH 2009. IEEE International Symposium on*, pages 3–7, Oct 2009.

[15] C. Bichlmeier, F. Wimme, S.M. Heining, and N. Navab. Contextual anatomic mimesis hybrid in-situ visualization method for improving multi-sensory depth perception in medical augmented reality. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, pages 129–138, Nov 2007.

[16] T. Akinbiyi, C.E. Reiley, S. Saha, D. Burschka, C.J. Hasser, David D. Yuh, and A.M. Okamura. Dynamic augmented reality for sensory substitution in robot-assisted surgical systems. In *Engineering in Medicine and Biology Society, 2006. EMBS '06. 28th Annual International Conference of the IEEE*, pages 567–570, Aug 2006.

[17] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, March 2010.

[18] John Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, Nov 1986.

[19] M.A. Ruzon and C. Tomasi. Color edge detection with the compass operator. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 2, pages –166 Vol. 2, 1999.

[20] Marco Balduini, Irene Celino, Daniele DellAglio, Emanuele Della Valle, Yi Huang, Tony Lee, Seon-Ho Kim, and Volker Tresp. Bottari: An augmented reality mobile application to deliver personalized and location-based recommendations by continuous analysis of social media streams. *Web Semantics: Science, Services and Agents on the World Wide Web*, 16(0):33 – 41, 2012. The Semantic Web Challenge 2011.

[21] Mark A. Ruzon and Carlo Tomasi. Edge, junction, and corner detection using color distributions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 23(11):1281–1295, November 2001.

[22] Shahid Ayub, Alireza Bahraminasab, and Bahram Honary. *A Sensor Fusion Method for Smart phone Orientation Estimation.* 2012.

[23] JeffreyR. Blum, DanielG. Greencorn, and JeremyR. Cooperstock. Smartphone sensor reliability for augmented reality applications. 120:127–138, 2013.

[24] Smartphone rotational axis. `http://it.mathworks.com/hardware-support/android-sensor.html`.

[25] Sensor functioning. `http:www.st.com/web/catalog/sense_power/FM89/SC1288/PF252443?sc=internet/analog/product/252443.jsp#`.

[26] Egnos system. `http://egnos-portal.gsa.europa.eu`.

[27] Homeria. `http://www.homeria.com/`.

[28] P. Chippendale, M. Zanin, and C. Andreatta. Spatial and temporal attractiveness analysis through geo-referenced photo alignment. In *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, volume 2, pages II–1116–II–1119, July 2008.

[29] A.K. Dubey and S.K. Shandilya. Exploiting need of data mining services in mobile computing environments. In *Computational Intelligence*

and Communication Networks (CICN), 2010 International Conference on, pages 409–414, Nov 2010.

[30] Otávio A. B. Penatti, Eduardo Valle, and Ricardo Torres. Comparative study of global color and texture descriptors for web image retrieval. *Journal of Visual Communication and Image Representation*, November 2011.

[31] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[32] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

[33] Wikipedia. Cross-correlation — Wikipedia, the free encyclopedia, 2013.

[34] Christoffer Valgren and Achim J. Lilienthal. Sift, surf & seasons: Appearance-based long-term localization in outdoor environments. *Robot. Auton. Syst.*, 58(2):149–156, February 2010.