A Framework for Outdoor Mobile Augmented Reality and its Application to Mountain Peak Detection

Roman Fedorov, Darian Frajberg, and Piero Fraternali

Politecnico di Milano,
Dipartimento di Elettronica, Informazione e Bioingegneria
Piazza Leonardo da Vinci, 32, Milan, Italy
`{first.last}@polimi.it`

Abstract. Outdoor augmented reality applications project information of interest onto views of the world in real-time. Their core challenge is recognizing the meaningful objects present in the current view and retrieving and overlaying pertinent information onto such objects. In this paper we report on the development of a framework for mobile outdoor augmented reality application, applied to the overlay of peak information onto views of mountain landscapes. The resulting app operates by estimating the virtual panorama visible from the viewpoint of the user, using an online Digital Terrain Model (DEM), and by matching such panorama to the actual image framed by the camera. When a good match is found, meta-data from the DEM (e.g, peak name, altitude, distance) are projected in real time onto the view. The application, besides providing a nice experience to the user, can be employed to crowdsource the collection of annotated mountain images for environmental applications.

Keywords: Outdoor augmented reality, mobile, real-time, mountain peak identification, environment monitoring, computer vision

1 Introduction

Outdoor augmented reality applications exploit the position and orientation sensors of mobile devices to estimate the location of the user and her field of view so as to overlay such view with information pertinent to the user's inferred interest. These solutions are finding a promising application in the tourism sector, where they replace traditional map-based interfaces with a more sophisticated user experience whereby the user automatically receives information based on what he is looking at, without the need of manual search. Examples of such AR apps include, e.g, Metro AR and Lonely Planet's Compass Guides[1]. The main challenge of such applications is providing an accurate estimation of the user's current interest and activity, adapted in real-time to the changing view. Commercial applications, which operate mostly in the tourism field, simplify the problem

[1] http://www.lonelyplanet.com/guides

by estimating the user's interest based only on the information provided by the device position and orientation sensors, irrespective of the content actually in view. Examples are sky maps, which show the names of constellations, planets and stars based on the GPS position and compass signal. An obvious limit of these approaches is that they may provide information that does not match well what the user is seeing, due to errors in the position and orientation estimation or to the presence of objects partially occluding the view. These limitations prevent the possibility for the AR application to create *augmented content* usable for monitoring purposes. If the overlay of the meta-data onto the view is imprecise, it is not possible for the user to save a copy of the augmented view, e.g., in the form of an image with captions associated to the objects. Such augmented content could be useful for several purposes: archiving the augmented outdoor experience, indexing visual content for supporting search and retrieval of the annotated visual objects, and even for the extraction of semantic information from the augmented content.

This paper describes the data management approach of *SnowWatch*, an outdoor mobile AR application for the automatic annotation of mountain peaks with geographical meta-data (peak name, altitude, distance from viewer, etc). Unlike other systems (e.g., PeakFinder), SnowWatch exploits a content-based reality augmentation algorithm, which takes in input not only the position and orientation of the user's device but also the content of the current view and a Digital Elevation Model (DEM), which is a 3D representation of the Earth's surface stored at the server side. First, the DEM, the position and the orientation of the user are exploited to estimate a bi-dimensional projection of the panorama that should be viewed by the camera of the mobile device and to match such virtual panorama to the image currently captured by the camera. Second, meta-data about mountain peaks are transferred from the DEM to the camera view; they are superimposed to the camera view, so that the user can save an augmented image of the mountain landscape, which integrated the contextual meta-data and the view captured by the user. The augmentation process must be performed in real-time, which requires a fast processing and transmission of the DEM data, which constitute the largest portion of the data exchanged between the application server and the mobile terminal.

The contributions of the paper can be summarized as follows:

- We introduce the problem of reality augmentation, specifically for mountain landscape views.
- We summarize our previous results in the offline detection of mountain peaks in static, geo-referenced images.
- We highlight the challenges of porting the offline algorithms to a mobile AR context, in terms of accuracy, stability of the registration of the camera view to the virtual panorama, and unreliable network connectivity.
- We describe a framework for the development and testing of outdoor mobile AR applications created for addressing the above-mentioned challenges.
- We illustrate the application of the framework to real-time peak detection and different optimization techniques that have been introduced in the generic framework to support the mountain peak identification task.

- We define a performance evaluation metrics based on a scalar value simulating the error that would be perceived in a real usage session.
- We report on the preliminary results of evaluating the SnowWatch app in real outdoor experimental conditions.

The rest of the paper is organized as follows: Section 2 overviews previous work in the areas of outdoor augmented reality applications, mountain image analysis, and environmental monitoring applications; Section 3 states the problem of outdoor AR application design and presents a generic architectural framework addressing the challenges of this class of applications; Section 4 shows the application of the framework to real-time mountain peak detection: it sets the background of the problem, briefly recaps our previous results for *offline* peak identification, highlights the challenges of the real-time AR version, and discusses the optimization techniques implemented, reporting the preliminary results of their evaluation; Section 5 concludes by presenting the outcome of using annotated mountain images for the resolution of a real-world environmental problem, and provides an outlook about the next research objectives.

2 Related work

Augmented reality applications. AR is a well established research topic within the Human Computer Interaction field, which has recently attracted new attention due to the announcement by major hardware vendors of low-cost, mass-market AR devices. In particular, the recent trend of mobile devices as AR platforms benefits from the improved standardization (most AR software can now be used without ad hoc hardware), increased computational power and sensor precision [13]. The survey in [3] overviews the history of research and development in AR, introduces the definitions at the base of the discipline, and positions it within the broader landscape of other technologies. The authors also propose design guidelines and examples of successful AR applications and give an outlook on future research directions. An important branch of the discipline is the outdoor AR. Several works address the problem, usually to identify [5] and track [21] points of interest in urban scenarios. Although standard solutions for mobile AR already exist (e.g. Wikitude[2]), they rely only on compass sensors or the a priori known appearance of the objects. We present a novel framework for the fusion of the two techniques: refining the compass-based AR performance without knowing a priori the appearance of the objects.

Mountain image analysis. Image analysis in mountain regions is a well investigated area, with applications that support environmental studies on climate change and tourism [7]. Mountain image analysis research focuses on peak identification in public photographs [2, 1] and the problem of segmenting the portion of the photograph corresponding to a certain mountain in snow covered areas [24, 22]. A prominent application field of mountain image analysis is snow information extraction. Traditionally snow is monitored through manual

[2] http://www.wikitude.com/app/

measurement campaigns, permanent measurement stations, satellite photography, and terrestrial photography. Most approaches (e.g. [24, 22]) rely on cameras designed and positioned ad hoc by researchers, and are not applicable to user-generated images created in uncontrolled conditions. Porzi et al. [18] propose an app for mountain peak detection; the contribution focuses on the time efficient peak identification and does not address mobile AR requirements, such as real-time response, asynchronous dynamics of the algorithms and uncertain internet connection. Furthermore, the authors report the performance of the algorithms in terms of error vs time measures, so it is unclear which of the algorithms would provide a better user experience. In our work we propose a Capture and Replay testing framework that provides a unique performance measure capturing exactly the error that would be perceived by the user. SnowWatch has the potential of enabling a novel generation of mountain environment monitoring applications, in which the augmented images created by the users during tourist trips are reused for extracting information useful for environment management and planning problems. We report our first results in this direction in Section 5.

Environmental citizen science applications. "Citizen science" refers to the direct engagement of non-specialized people (the citizens) to help address scientific problems [15]. The massive diffusion of social media, with its powerful tools for public communication, engagement, and content sharing, has multiplied the ways to engage volunteers and exploit relevant public User-Generated Content (UGC). In particular, social media combined with mobile devices favored the collection of *geo-located* UGC in applications related to spatial information, so-called Volunteered Geographical Information Systems (VGIS), in which citizens help enhance, update or complement existing geo-spatial databases [11]. Several approaches have been applied to disaster management for e.g., earthquake mapping [29] and rapid flood damage estimation [19]. Applications monitoring hazards through the collection of user-generated content are also reported: tweet distribution analysis for monitoring is employed in [23] for earthquakes and in [25] for floods. Examples exist of continuous monitoring applications in the environmental field: bird observation network [26], phenological studies [20], hydrological risk assessment [6], plant leaf status assessment [17] and geological surveys (`http://britishgeologicalsurvey.crowdmap.com`). Besides text, also visual content, such as Flickr photographs [27] and public touristic webcams [16] have been used to monitor environmental phenomena, such as coarse-grained snow cover maps [27], vegetation cover maps [28], flora distribution [27], cloud maps [16] and other meteorological processes [12].

3 A Framework for Mobile Outdoor AR applications

The problem addressed in this work is the design of mobile AR applications for the enrichment of outdoor natural objects. Restricting the focus to devices that support a bi-dimensional view, a generic architecture must be realized that receives as a first input a representation of the reality - in which the user is embedded - captured by the device sensors; such representation typically comprises a sequence of camera frames captured at a fixed rate, and the position and orientation of the device, captured by the GPS and orientation sensors; the second input is the information about the possible objects present in a region of interest. The output is the on-screen position of relevant objects and the association of relevant meta-data to such objects, computed at the same frequency of the input capture. Besides the near real-time execution time, the system must also cope with the following requirements:

- *Uncontrolled viewing conditions*: the objects to be identified have no fixed, known a priori, appearance, because the viewing conditions can drastically change due to weather, illumination, occlusions, etc.
- *Uncertain positioning*: position and orientation sensor errors make the location estimation potentially noisy; thus the identification of the relevant objects from these signals alone cannot be assumed to be fully reliable and must be corrected with information from the camera view.
- *Bi-dimensional reduction*: although the objects' position in the real world is estimated in the 3D space, the on-screen rendition requires a projection onto the 2D surface of the camera view, based on a model of the camera.
- *Uncertain internet connection*: especially for rural and mountain regions.

Figure 1 shows a representation, through an UML component diagram, of the reference architecture of a mobile outdoor AR application. The key idea is to enable the near real-time reality augmentation process thanks to a proper partition of functionality and a mix of synchronous and asynchronous communications among the modules. The architecture consists of four sub-systems: the Sensor Manager, the Data Manager, the Position Alignment Manager and the Bi-dimensional Graphical User Interface, which draws objects and their meta-data in provided on-screen coordinates.

3.1 Sensor Manager

The *Sensor Manager* coordinates data acquisition from the device sensors. It typically comprises one module per each signal processed by the application; the typical configuration comprises the GPS Sensor Manager, the Orientation Sensor Manager and the Camera Sensor Manager. The modules work asynchronously and provide input to the Position Alignment Manager and Data Manager, which subscribe to their interface and are notified when a new signal arrives from a sensor.

3.2 Data Manager

The *Data Manager* is responsible for providing to the other sub-systems the initial positions of the objects in view and the meta-data for enriching them. It receives as input the specification of an area of interest (typically, inferred from the user's position, which defines the region the user may be looking at, or may be moving within), and interacts with an external repository containing a virtual

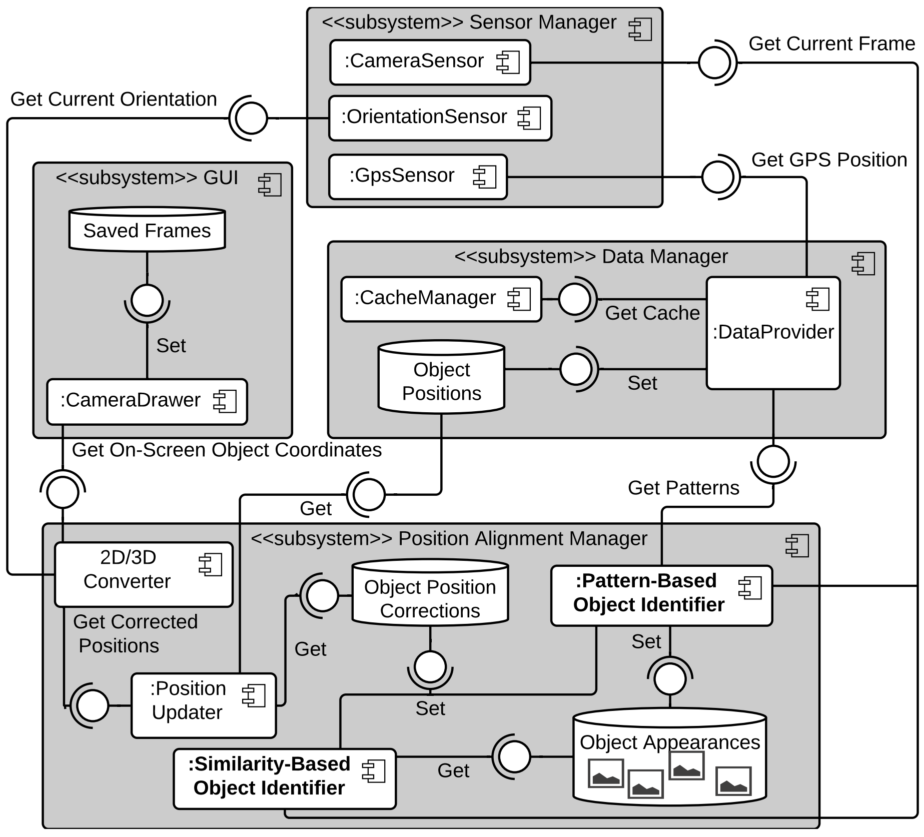

Fig. 1. The proposed architecture of a mobile outdoor AR application.

representation of the world (e.g, a sky map or a DEM). It produces as output *Object Positions*, which specify the (initially approximate) 3D coordinates of the candidate objects to display. Within the *Data Manager*, a *Data Provider* component queries one or more external geo-referenced data sources, with the current user's location, and extracts the coordinates of the objects that are likely to lie within the view of the user. For example, in a sky observation app, it queries the sky map for the celestial coordinates, plus meta-data such as type, name, distance, etc., of the potentially visible objects. The *Cache Manager* implements data pre-fetching and synchronization policies, based on information about current cache content, network availability, and cost of data transfer. Since data about the objects can be large the Cache Manager realizes a trade-off between on-demand transfer from external data sources and caching in the local storage of the device. Furthermore, it enables disconnected usage, as needed in the outdoor scenario, in which internet connection may not be always granted.

3.3 Position Alignment Manager

The Data Manager provides a *fast* computation of the initial Object Positions, to enable the immediate update of the GUI. But its output may be noisy, because the estimated user's position, the camera orientation and the virtual world representation may all contain errors. It is well-know that the GPS and orientation signal of mobile devices may be inaccurate; on the other hand, also the virtual world representation, e.g., a DEM describing the earth surface, may be affected by errors, e.g., due to low resolution. Therefore, the Position Alignment Manager comprises components for updating the positions of the objects, adapting them to the actual content of the camera view, and projecting them to the device's view. It takes in input the initial object positions provided by the Data Manager and produces in output the *corrected* on screen object coordinates. To support the trade-off between accuracy and speed, the (demanding) computations required for improving accuracy are delegated to separate modules, which provide asynchronous corrections to the initial candidate positions, by applying content-based object detection techniques. These modules feed the *Object Position Corrections* store (see Figure 1) with the adjustments computed asynchronously, which the *Position Updater* and *3D/2D Converter* components exploit to correct the on screen coordinates used by the GUI. Examples of components for the content-based refinement of object positions are *Pattern-Based* and *Similarity-Based* Object Identifiers.
A *Pattern-Based Object Identifier* performs a frame-based match. It uses the virtual world representation as a pattern to search within the real world image. It takes in input the virtual representation of the world (e.g., the synthetic rendition of a constellation or of a piece of mountain skyline) and computes a ranked list of approximate matches between the virtual image and the real one, with respect to some similarity function. As a collateral output, the Pattern-Based Object Identifier can also extract from the real world image the regions that correspond to the identified objects, according to the best match. Such artifacts, cached in the *Object Appearance Store* of Figure 1, denote the visual appearance of the objects of interest in the current view and can be used for accelerating the correction of objects' positions when the view changes.
A *Similarity-Based Object Identifier* performs object-based similarity search; it takes in input the object appearance artifacts and searches them in the frame, using computer vision techniques.
Finally, the *2D/3D Converter* projects 3D positions onto the bi-dimensional screen space. It takes in input the device position, orientation, and Field Of View (FOV), applies a prospective projection, determines the on-screen coordinates of the candidate objects and discards those out-of-view, e.g, due to micro-movements of the device. For example, it projects the celestial coordinates of the relevant sky objects into on-screen coordinates. The on-screen coordinates are used by the GUI for rendering the augmented reality view.
The asynchronous communication between the components that compute position corrections and those that project positions and render the virtual reality view aims at enabling a best effort, near real-time adjustment of the view. The

prospective projection is a constant-time procedure, so that the total response time of the Position Updater and of the 3D/2D Converter is linear w.r.t. to the number of candidate objects. Since this number is reasonably bound, the resulting time complexity is constant, which allows the mobile device to call the Position Updater and the 3D/2D Converter *synchronously* at every frame arrival and redraw the view in near real-time based on the best available approximation of the object positions.

3.4 Capture and Replay Testing framework

Testing an outdoor AR application is a complex task that requires evaluating simultaneously the precision of object positioning and the response time, two competing objectives, in a realistic setting that considers the sensor inputs (not available in the lab). The assessment criteria must also take into account usage conditions: if the user keeps the device steady, low error is the prominent goal, while higher execution time due to re-positioning after micro-movements is less relevant; conversely, if the device is subject to movement (e.g, during walking), fast execution can be more important than object positioning precision. Therefore, testing should be supported by an auxiliary architecture that helps achieve the following objectives:

– Perform lab testing in conditions equivalent to real outdoor usage.
– Contrast different designs in the same operating conditions and assess the same designs under different operating conditions.
– Use the performance metrics best suited to a specific application and operating condition.

To support such requirements, we have extended the architecture of Figure 1 with a testing framework based on a *Capture & Replay* approach:

– A *Capture application*: it is a mobile application that can be used to record an outdoor usage session, complete with all sensor data (camera, GPS and orientation) and user's activity (start, stop, video record, snapshot, etc.).
– An *Annotation application*: it is an application that allows one to annotate the frames of a usage session with the position of the visible objects, so to create a gold standard for evaluating the accuracy of object positioning.
– A *Replay test driver*: it is an application that can attach to the Position Alignment Manager sub-system of the architecture of Figure 1 and measure its performance based on a plug-in metrics.

The Capture application collects execution traces. A trace consists of a sequence of entries that record all the events occurred during a usage session, including: information about the device manufacturer and model; the set of frame images taken at frequency F, with their acquisition timestamp; and the sequence of time-stamped sensor readings, i.e., the values of the position and orientation sensors acquired at the maximum frequency supported by the device. The above mentioned information, logged by default, is normally sufficient to reproduce the user activity for a typical outdoor AR application; however, the Capture application can be extended to support additional logging, if needed by a specific application. The Annotation application allows the developer to associate with each frame zero or more triples $< i, x_i, y_i >$, where i is the index of an object visible on that frame, and x_i, y_i are the on-screen coordinates of that object. The annotated frame sets can be used as the ground truth for the assessment of the Position Alignment Manager. The Replay test driver is an application stub that replaces the Sensor Manager, Data Manager and GUI of the architecture of Figure 1, so to reproduce the sequence of events and sensor readings previously recorded by the Capture application. It takes in input one or more traces and supplies to the Position Alignment Manager the frame images, at the capture frequency F, and the corresponding values of the sensor readings; then, it retrieves from the Position Alignment Manager the estimation of on-screen object positions and evaluates them w.r.t. the ground truth and to a selected metrics. The Replay test driver utilizes a metric function (called Real-Time Average Angular Error, RTAAE) that considers the positioning errors of all the relevant objects. For each visible object $i = 1, \ldots, n$ let (x_i, y_i) be the on-screen coordinates predicted by the Position Alignment Manager, while $(\hat{x_i}, \hat{y_i})$ be the ground truth coordinates. We define the angular error in the position of the i-th object as

$$\varepsilon(\hat{x_i}, \hat{y_i}) = \sqrt{d_x(\hat{x_i}, x_i)^2 + d_y(\hat{y_i}, y_i)^2},$$

where

$$d_x(\hat{x}, x) = min(360 - \frac{f}{w}|\hat{x} - x|, \frac{f}{w}|\hat{x} - x|)$$

is the angular distance (in degrees) between the predicted and ground truth coordinate along the azimuth axis, given the circular symmetry, f is the horizontal FOV (in degrees) of the camera and w is the width (in pixels) of the image. Similarly we define the angular distance along the roll axis:

$$d_y(\hat{y}, y) = \frac{f}{w}|\hat{y} - y|$$

Note than the same angular resolution in degrees/pixel is assumed for both axes, because the elevation angles are small. The angular error for an entire sequence can be defined as the average angular error of each frame. Finally, given N traces, the Real-Time Average Angular Error (RTAAE) is the average over all traces.

Note that the described Capture & Replay approach allows lab tests to assess the Position Alignment Manager in the same operating conditions that occur in an outdoor session, because it exploits the same frame acquisition rate and sensor sampling frequency experimented in the real time use.

4 SnowWatch: an outdoor AR application for Mountain Image Enrichment

This section describes how the architecture of Figure 1 has been adapted to the development of a mobile AR application for real-time mountain peak de-

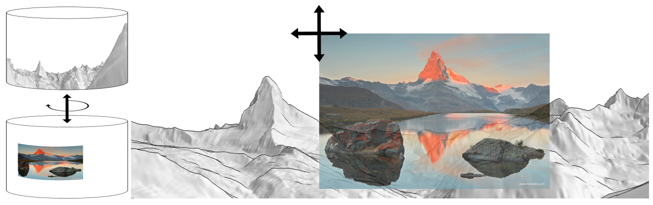

Fig. 2. Photo to panorama cylindrical and equivalent 2D Cartesian alignment.

tection, by porting existing algorithms developed for the offline identification of peaks in geo-tagged photographs to the mobile AR context[3]. Here, we highlight the challenges in adapting the algorithms to the mobile AR scenario, discuss the application-specific solutions, and report preliminary evaluation results obtained with the testing framework described in Section 3.4. The offline algorithms have been developed in the SnowWatch project [9][4], which tackles mountain monitoring with a Citizen Science application for the collection of public Alpine images and the extraction of snow indexes usable in water availability prediction models. To this aim, SnowWatch crawls a large number of images from content sharing sites and touristic webcams, classifies those images that portrait mountain peaks and contain the location of shooting, identifies visible peaks by automatically aligning each image to a synthetic rendition computed from a public DEM, finds the pixels of each peak that represent snow and calculates useful snow indexes (e.g, minimum snow altitude). These indexes are then used to feed existing water prediction models and compared with other official sources of information.

The SnowWatch Web architecture is mainly server-side and thus very different from the mobile AR architecture of Figure 1: it combines data providers (photograph crawler and webcam crawler), data consumers (front-end web portal and environmental models) and back-end processes that analyze photographs and enrich them with landscape and environmental meta-data (orientation of the photograph, mountain peak positions, snow covered areas). The interaction of the user is similar to that of a sharing site: uploading one's photos, applying filters (for peak detection, in this case), correcting the position of peaks manually, and rating, sharing, and commenting the resulting pictures.

4.1 Offline Peak Detection for the Web

One of the key algorithms of SnowWatch Web architecture is the offline peak identification. Peak positions are obtained through the alignment between the photo and the terrain model. Given a photograph and the meta-data extracted from its EXIF container (geo-tag, focal length, camera model and manufacturer), a matching is performed with a 360° panoramic view of the terrain synthesized from a public, Web-accessible DEM. The rendered panorama contains

[3] A detailed description of the offline algorithms can be found in [8, 10]

[4] http://snowwatch.polimi.it/?lang=en

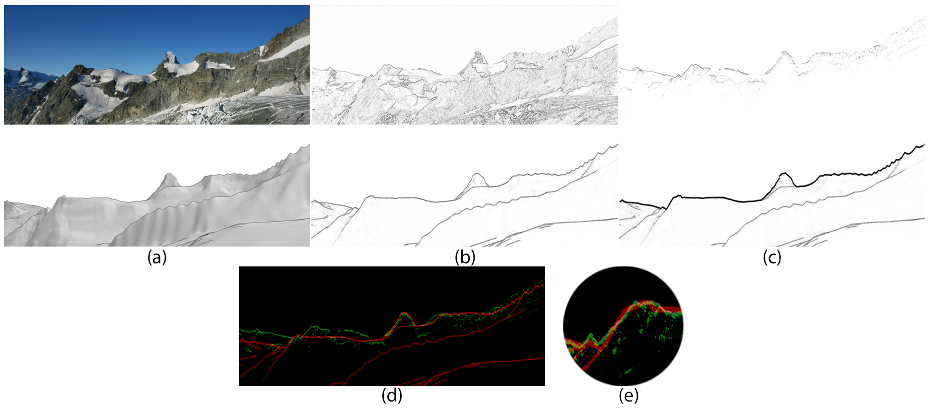

Fig. 3. An example of the photo-to-terrain alignment: (a) input photograph (top) and corresponding panorama (bottom), (b) edge extraction, (c) skyline detection, filtering and dilation (d) global alignment with refinement (e) local alignment.

the mountain peak positions, so once a correct overlap is found, peak positions are projected from the panorama to the photo. The alignment can be seen as the search for the correct overlap between two cylinders (assuming the zero tilt of the photograph): one containing the 360° panorama and the other one containing the photo, suitably scaled. As Figure 2 shows, this is equivalent to look for the offset between the photo and the unfolded 2D panorama that guarantees the best overlap. The alignment method proceeds in four steps, described below and illustrated in Figure 3.

Preprocessing: The horizontal Field Of View (FOV) of the photograph is calculated from the focal length and the size of the camera sensor. Then, the photograph is rescaled considering that the width of the panorama corresponds to a FOV equal to 360°. After this step, the photo and the panorama have the same scale in degrees per pixel and thus matching can be performed without the need of scale invariant methods. Then an edge extraction algorithm is applied to both the photograph and the panorama to produce an edge map, which assigns to each pixel the strength of the edge at that point and its direction (Figure 3b).

Matching edges of an image with those of a virtual panorama requires addressing the fact that there is not a one-to-one mapping between edge pixels extracted from the two sources. The photo generates many noisy edges that do not correspond to the mountain slopes, but to other objects in the foreground (e.g., rocks, trees, lakes, houses, etc.) and in the background (e.g., clouds, snow patches, etc.). Thus, a skyline detection algorithm is employed [14], and all the edge pixels above the skyline are removed, being considered obstacles or clouds. Then, a simple weighting mechanism is applied, which assigns decreasing weights to the edge pixels as the distance from the skyline increases (Figure 3c - top). As for the panorama, the edges corresponding to the skyline can be simply identified as the upper envelope of the edge map, by keeping, for each column of pixels, the topmost edge point. Since the edge filtering of the photograph emphasizes the edges of the skyline, a morphological dilation is applied to emphasize the edges corresponding to the skyline of the panorama (Figure 3c - bottom).

Global alignment: The matching between the photograph and the corresponding panorama is performed using a Vector Cross Correlation (VCC) technique [2], which takes into account both the strength and the direction of the edge points. The output of the VCC is a correlation map that, for each possible horizontal and vertical displacement between the photograph and the panorama, indicates the strength of the matching.

Local alignment: to improve the precision of the position of each mountain peak, a local optimization is applied. For each peak we consider a local neighborhood centered in the photograph location identified as the peak position by the global alignment. In this way each peak position is refined by identifying the best match in its local neighborhood. Overall, this is equivalent to applying a non-rigid warping of the photograph with respect to the panorama.

4.2 Mobile Peak Detection for AR apps

The mobile version of the mountain peak detection task requires significant adaptations of the offline approach, to comply with the architecture of Figure 1. The main challenges induced by the mobile and real-time AR requirements include:

- Lower computational power w.r.t. Web multi-tier architectures.
- Higher accuracy: while it is tolerable for a web-based application to misidentify mountain peaks (the image will be discarded, or manually adjusted by the Web user), an erroneous peak identification on a mobile application used live produces a disappointing user experience and the enriched image, once saved, can not be easily fixed on a small screen device.
- Faster response time: peak positions must be overlaid in real-time and no overhead for image processing initialization is acceptable, because mobile users do not tolerate delays in the order of seconds at app every start.
- Data storage and transfer: the whole data set for peak identification (the DEM) is too big to be stored entirely in the a mobile device; at the same time, the internet connection for downloading the needed data on the fly, can not be assumed always available. Indeed, a mountain peak AR app must be usable in mountain regions, where even today internet coverage is patchy.
- Technical constraints: mobile application development imposes numerous restrictions on the supported architectures, frameworks and libraries.

On the other hand, a significant advantage of the mobile version is the availability in real-time of the position and orientation sensor values, which, although subject to error, provides an estimate of the panorama in view.

The SnowWatch mobile AR application specializes the architecture of Figure 1. In the sequel, we describe the application-specific concepts and component refinements introduced for the mobile context.

The *objects* to be identified are mountain peaks and the *object positions* are 3D global system coordinates laying on a unit sphere centered in the device location.

An application-specific *Cache Manager* has been implemented, responsible for pre-fetching and caching the DEM fragments corresponding to the geographical region the user is visiting. Pre-fetching is enabled when the WiFi connection of the device is on and cache data are used by the DataProvider component to compute the Object Positions during outdoor usage. When the user moves out of the region for which data are in the cache, a cache miss triggers the download of a new fragment, which, in case of cache full, replaces the fragment relative to the region visited earliest.

The *Similarity-Based Object Identifier* component is implemented with a state-of-the-art cross-correlation patch recognition technique [4], which has been ported to the mobile execution environment.

The component where the most relevant adaptations have been introduced is the Pattern-Based Object Identifier, described next.

4.3 Pattern-Based Object Identifier

The Pattern-Based Object Identifier implements the pattern matching between the skyline extracted from the DEM and the skyline visible in the camera view, and computes Object Position Corrections based on the outcome of such procedure. It has been realized starting from the experience for offline peak detection described in Section 4.1, introducing significant improvements.

Non-Zero Tilt. The web version of the matching algorithm assumes the camera tilt as negligible (equal to 0) and reduces the problem to the alignment between two cylinders, avoiding the (much more costly) spherical match. This assumption proved viable experimentally; mountain ranges are far from the position of the user and the error induced by a moderate tilt is compensated by the skyline matching algorithm. On a mobile device the assumption of zero or constant tilt must be relaxed, to cope with the movements of the mobile device made by the user during a viewing or shooting session. To avoid switching from 2D cylindrical to 3D spherical alignment, which would jeopardize the response time, we designed an approximate approach: the input image is rotated by the tilt provided by the orientation sensor, standard 2D alignment is performed, and the final peak coordinates are rotated in the inverse direction at the end. This method deals with tilting effectively and preserves the fast response time of the 2D alignment, to obtain corrections to the 3D object positions.

Edge Filtering and Skyline Detection. The heuristic methods described in Section 4.1 work well for offline peak detection, because they are applied to pre-filtered images (fixed webcams have a view that does not change and can be manually checked once and for all for suitability; user generated photos go through an offline binary classification step to retain only samples with obstacle-free skyline view). But they are not well suited to a mobile AR scenario, where it is more likely that the camera is used in adverse weather conditions and in presence of transient occlusions of the skyline. In these cases, a cloud, a high voltage

cable, or a roof would be recognized as part of the mountain skyline; this would impact the heuristic edge filtering, e.g., a cloud edge would be treated as skyline and the mountain slope below it would be considered as noise. Such erroneous classification would hamper the alignment with the DEM and the positioning of peaks, yielding an unacceptable user's experience.
To increase robustness even to small, transient occlusions, we developed a new approach, based on the application of a Convolutional Neural Network (CNN) supervised learning algorithm, which finds the *landscape skyline*, i.e., the set of all points that represent the boundary between terrain slopes and the sky. First of all, a simple and fast Canny edge detector extracts a draft binary edge map. Then every pixel of such map is classified as positive or negative, where positive means that it belongs to the landscape skyline. The edge pixels classified negatively are removed from the edge map. For each edge pixel a $K \times K \times 3$ RGB patch centered at the pixel coordinates is extracted and classified with an image content-based classifier. The choice of the CNN over other machine learning algorithms (e.g. Logistic Regression, SVM, Random Forest) is motivated by the ability of the CNN to learn the best features to employ, which avoids their manual, and subjective, definition. Conversely, a typical downside of using CNN, the need of a very large amount of training data, is not an obstacle in our case, because the items to classify are small patches extracted from the neighborhood of image edges; in our experiments, an average 640×480 outdoor image contains tens of thousands of edge pixels. To build the training and test sets, it is sufficient to manually annotate the image with its landscape skyline. Then, all the patches corresponding to the edge points can be extracted and classified automatically (positive if the center is located no more than d pixels from a skyline point). With this semi-automatic procedure, it is possible to generate the massive amount of training data necessary to train the CNN, with low effort.
Figure 4 shows an example of alignment taken in very adverse conditions: the input image (top left) is taken from behind a window, the corresponding fragment of the panorama (middle left) contains two mountain peaks (red arrows). The edges extracted from the input image (top center) contain an enormous amount of noisy edges (mountain vegetation, houses, window frame) that would make the alignment with the panorama impossible; the CNN filtering procedure (top right, green points) successfully retains only skyline edge pixels. The panorama skyline to match is extracted simply by picking top points (middle center, red points); the alignment between the two skylines (middle right) allows us to project the two peak positions on the input image with high precision (bottom, augmented image). This resulted is computed in real time and the peak positions remain precise even when the user tilts or moves the mobile phone.

Occlusion Management The virtual panorama view contains only the peaks that could be visible by an observer based on the elevation model; in the real image, virtually visible peaks can be occluded by irrelevant objects, such as houses, people or even clouds or fog. The CNN network used for edge filtering in the mobile AR scenario helps dealing with occlusions: the network is trained to recognize the *landscape skyline*, i.e., the portion of the topmost edges that

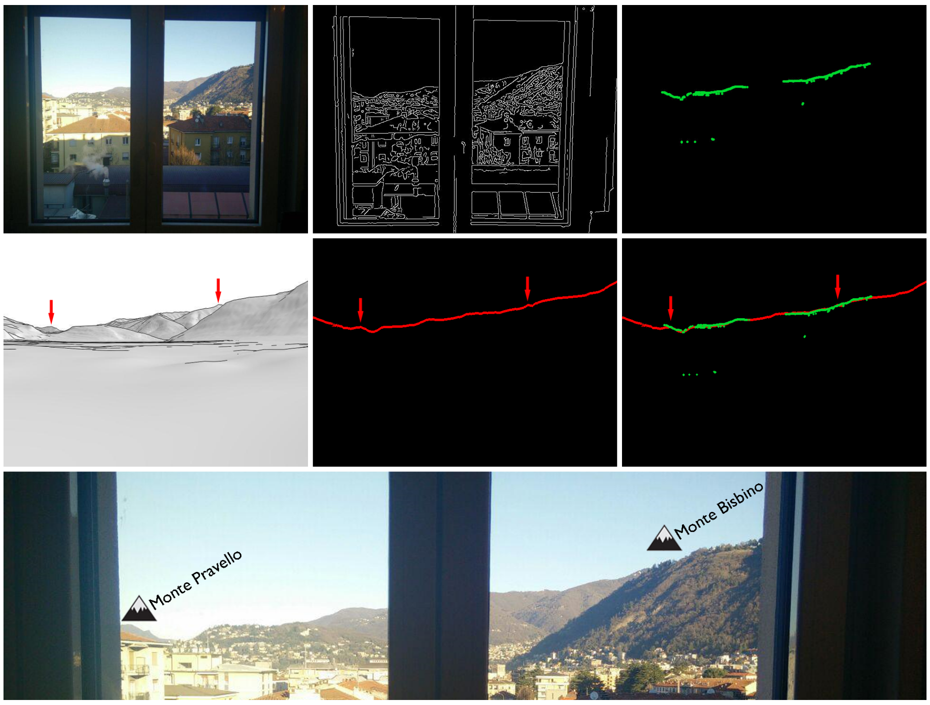

Fig. 4. Example of the peak identification in presence of many noisy edges.

actually represent the boundary of a mountain slope. This capability supports effective occlusion detection. Given a correct alignment between the landscape skyline of the image and the virtual skyline of the panorama, the peaks that are actually visible in the image will have fragments of the landscape skyline in their vicinity, while occluded peaks will not. Thus, once the alignment is found, for each peak a visibility score v is defined as the number of landscape skyline points located no farther than d pixels from the peak position. A peak is considered visible if $v \geq \bar{v}$ (where $\bar{v}$ is a fixed threshold). If a peak is considered visible, its appearance patch is extracted and cached; otherwise no patch is extracted (or its patch is removed from the cache, if previously stored). In this way, the Similarity-Based Peak Identifier will not find the patch inside the future frames. Figure 5 shows an example of peak identification with 3 virtually visible peaks. In this case, peak n.2 is occluded by the bell tower; indeed, besides a few false positive pixels, the bell tower contour is absent in the overall identified landscape skyline (top right). After the alignment, the neighborhood of each peak is analyzed (bottom right): peaks n.1 and n.3 present a large number of landscape skyline points (green dots) in their vicinity, while peak n.2 does not, so it is marked as non-visible and not included in the augmented image (bottom left).

Sensor Orientation The sensed orientation of the device can be used to improve the performance of the object identification. Since the match between

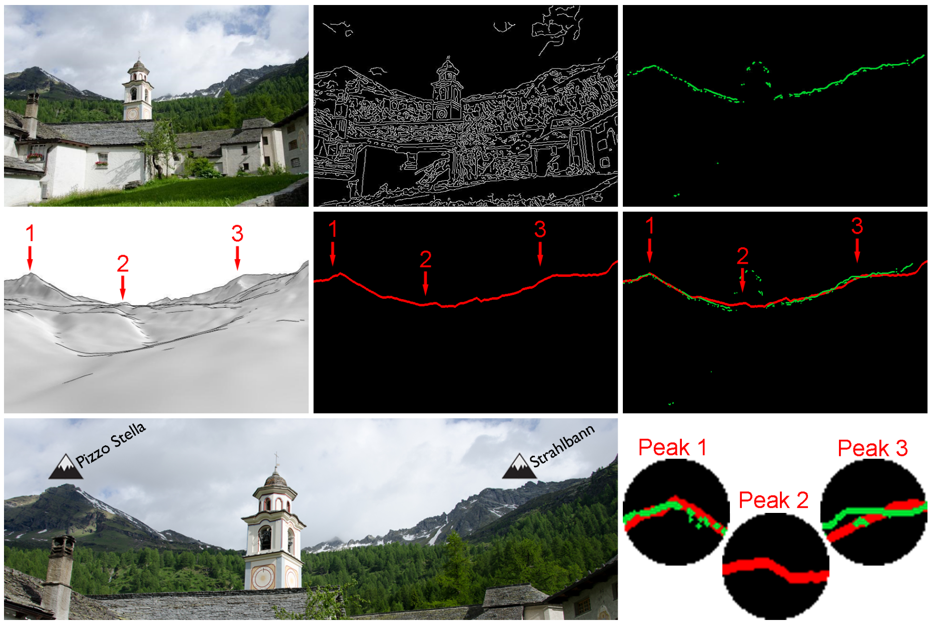

Fig. 5. Example of the peak identification in presence of occlusions.

the virtual panorama and image skyline is approximate, each candidate peak position receives a score, which is an estimate of the confidence of the match algorithm. Such score can be manipulated to take into account the agreement between orientation as sensed from the compass and estimated by the Position Alignment Manager. For example, a kernel function based on the difference between the sensed and estimated orientation can be used as a scale factor. Furthermore, the computation of peak alignment can be avoided in the areas of the image in which the kernel factor is equal to zero, because those regions would provide an unreliable peak position estimation. Such optimization decreases the computation time: we assume a maximum 15° orientation sensor error and perform the photo-to-panorama alignment not in the whole 360° panorama, but only in a $30° + FOV$ portion of it.

4.4 Experiments

This section presents the preliminary results of evaluating the landscape skyline detection algorithm and the accuracy of peak identification, in the mobile AR scenario. The experimental data set used for training the CNN comprises 158 mountain photos randomly crawled from Flickr and manually annotated with the landscape skyline. Out of these, 111 were included in the train set ($\sim 70\%$) and 47 in the test set ($\sim 30\%$). Then, for each image the binary edge map was computed, and patches (of size $28 \times 28 \times 3$) were extracted for each edge point and labeled as positive or negative. To guarantee the balance of the patches data set, the same number of positive and negative samples was extracted from each image (by random sub-sampling the larger class); to avoid over-emphasizing edge-rich images, a maximum of 400 positive and 400 negative patches were derived from each image. The splitting into test and train data sets was performed at the image level, and not at the patch level, to ensure a bias-free estimation of the ability of the classifier to adapt to scenarios not seen before (patches from the same image could not belong to both the training and test set). The overall observed accuracy of the CNN trained on the training set and evaluated on the test set was 96%, which resulted also in a satisfactory subjective judgment of the resulting skylines.

The evaluation of the peak identifier accuracy was performed on the *VENTURI Mountain Dataset* [18]. The data set is a collection of 12 outdoor sequences accompanied with GPS positions and orientation sensor logs, resulting in 3117 frames. For each frame the position of the mountain peaks is manually annotated. We measured the performance of the Pattern-Based Peak Identifier in terms of average peak position angular error. The observed average peak position error was 1.32°, which is lower than the minimum error obtained by the authors of [18] and defined suitable for mobile computation, namely 1.87°. The average time currently required by the pattern- and similarity-based peak identifiers to process a frame is respectively less than 3" and less than 1". Such times are totally dependent on the architecture and characteristics of the device being used, which in this case correspond to a Motorola Nexus 6 with Chipset Qualcomm Snapdragon 805, CPU Quad-core 2.7 GHz Krait 450, GPU Adreno 420, RAM 3GB, OS Android 5.1.1. On the other hand, due to the architecture of the system, the *on-screen peak positioning is always real-time* thanks to the sensor data, while the time complexity of peak identifiers influences only the update frequency of corrected peak positions.

Figure 6 shows an example of the mountain identification process in the mobile AR scenario. Initially, the on-screen peak positions are determined only through the orientation sensor data (top, red icons represent the predicted positions, arrows the real positions and the angular error is reported). After the photo-to-panorama alignment is performed, the peak positions are estimated more precisely (bottom left, green icons) and the corresponding mountain patches are extracted. The bottom right part of the figure shows how, in the next frames with a (slightly) different view, the same peaks can be quickly located by the similarity-based peak identifier.
Future experiments will collect a larger image data set, with occluded mountain skylines, which are missing in the Venturi data set. The described algorithms will be tested using the Capture and Replay framework of Section 3.4, to evaluate their response time with different mobile devices and RTAAE error.

5 Conclusions and Future Work

We have presented a framework for the development of outdoor mobile AR applications and discussed its use in SnowWatch, an application for mountain

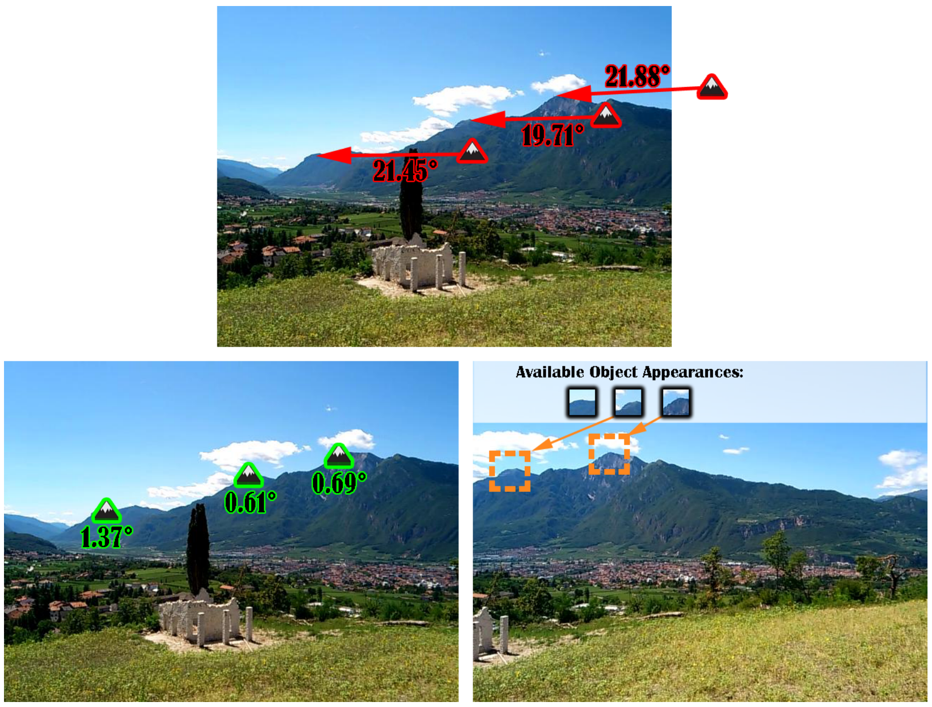

Fig. 6. Example of an sensor-, pattern- and similarity-based peak identifications. Images from the Venturi dataset [18] .

image enrichment. SnowWatch has the primary goal of attracting the interest of tourists, who could use it to enhance their outdoor experience with virtual expert knowledge about the mountain peaks in view. However, the project has also a second, equally important, goal: producing a repository of annotated mountain images for supporting environmental research. This requires assessing the environmental utility of information derived from public mountain images; to this end, we have collected a large set of images contributed by users and automatically crawled from touristic webcams[5] and extracted automatically snow information to address a water management problem in which snow is a determinant factor. In particular, we exploited a water management simulation model for the regulation of the Como Lake and evaluated the impact of adding snow-related indexes (e.g., minimum snow altitude) extracted from annotated mountain images to its input. Lake Como is a regulated lake in Northern Italy with an Alpine hydro-meteorological regime characterized by scarce discharge in winter and summer and water abundance in late spring and autumn due to rainfall and snow melt from catchment mountains. The lake inflow and effluent is the Adda River, which feeds hydroelectric power plants and serves five agricul-

[5] The data set is available at http://snowwatch.polimi.it

tural districts. The regulation model aims to support the decision makers in the daily setting of the lake level, so to prevent flooding in Como city, while ensuring water for agriculture. Farmers downstream would like to store water for the summer, but this increases the lake level and the flood risks. These competing goals generate a conflict between flooding and irrigation, which can be modeled using two quantitative objectives: 1. *Flooding*: the average annual number days when the lake is higher that the flooding risk threshold. 2. *Irrigation*: the daily average squared water deficit w.r.t. the daily downstream demand. Preliminary experiments with the two-objectives policy simulation model show that the virtual snow indexes computed from annotated mountain images help design more informed, and thus closer-to-the-optimum, water management policies. Specifically, the virtual snow indexes extracted from public mountain images have been compared with the official snow information of Region Lombardy, elaborated from ground stations and satellite data: even the snow information extracted from a single webcam stream in the lake catchment is capable of identifying policies with performance comparable to those conditioned on the official snow bulletin data. Our future plans aim at deploying the SnowWatch mobile app to the vast community of tourists and residents of the Lombardy region, to enlarge the mountain image data set and improve the predicting power of the mountain image information for lake regulation and for other environmental problems.

Acknowledgment. This work has been partially funded by European Commission and by the Lombardy Region through the PROACTIVE FESR Project[6] and by the FP7 CHEST Project[7] open call grant.

References

1. Baatz, G., Saurer, O., Köser, K., Pollefeys, M.: Large scale visual geo-localization of images in mountainous terrain. In: Computer Vision–ECCV 2012 (2012)
2. Baboud, L., Cadik, M., Eisemann, E., Seidel, H.P.: Automatic photo-to-terrain alignment for the annotation of mountain pictures. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 41–48. IEEE (2011)
3. Billinghurst, M., Clark, A.J., Lee, G.A.: A survey of augmented reality. Foundations and Trends in Human-Computer Interaction 8(2-3), 73–272 (2015)
4. Briechle, K., Hanebeck, U.D.: Template matching using fast normalized cross correlation. In: Aerospace/Defense Sensing, Simulation, and Controls. pp. 95–102. International Society for Optics and Photonics (2001)
5. Dähne, P., Karigiannis, J.N.: Archeoguide: System architecture of a mobile outdoor augmented reality system. In: null. p. 263. IEEE (2002)
6. Degrossi, L.C., Albuquerque, J., Fava, M.C., Mendiondo, E.M.: Flood citizen observatory: a crowdsourcing-based approach for flood risk management in brazil. In: 26th Int. Conf. on Software Engineering and Knowledge Engineering (2014)
7. Dizerens, C., Hüsler, F., Wunderle, S.: Webcam imagery rectification and classification: Potential for complementing satellite-derived snow maps over switzerland
8. Fedorov, R., Camerada, A., Fraternali, P., Tagliasacchi, M.: Estimating snow cover from publicly available images. IEEE Transactions on Multimedia PP(99) (2016)

[6] http://www.proactiveproject.eu
[7] http://www.chest-project.eu

9. Fedorov, R., Fraternali, P., Pasini, C.: Snowwatch: a multi-modal citizen science application. In: Web Engineering (2016)
10. Fedorov, R., Fraternali, P., Tagliasacchi, M.: 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 (2014)
11. Goodchild, M.F.: Citizens as sensors: the world of volunteered geography. GeoJournal 69(4), 211–221 (2007)
12. Hyvärinen, O., Saltikoff, E.: Social media as a source of meteorological observations. Monthly Weather Review 138(8), 3175–3184 (2010)
13. Jain, P., Manweiler, J., Roy Choudhury, R.: Overlay: Practical mobile augmented reality. In: Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services. pp. 331–344. ACM (2015)
14. Lie, W.N., Lin, T.C.I., Lin, T.C., Hung, K.S.: A robust dynamic programming algorithm to extract skyline in images for navigation. Pattern Recognition Letters 26(2), 221 – 230 (2005)
15. Memarsadeghi, N.: Citizen science [guest editors' introduction]. Computing in Science Engineering 17(4), 8–10 (July 2015)
16. Murdock, C., Jacobs, N., Pless, R.: Webcam2satellite: Estimating cloud maps from webcam imagery. In: Applications of Computer Vision (WACV), 2013 IEEE Workshop on. pp. 214–221. IEEE (2013)
17. NatureServe, F.: Natureserve explorer: an online encyclopedia of life (2012)
18. Porzi, L., Buló, S.R., Valigi, P., Lanz, O., Ricci, E.: Learning contours for automatic annotations of mountains pictures on a smartphone. In: Proceedings of the International Conference on Distributed Smart Cameras. p. 13. ACM (2014)
19. Poser, K., Dransch, D.: Volunteered geographic information for disaster management with application to rapid flood damage estimation. Geomatica (2010)
20. Reddy, S., Shilton, K., Burke, J., Estrin, D., Hansen, M., Srivastava, M.: Evaluating participation and performance in participatory sensing. UrbanSense08 p. 1 (2008)
21. Reitmayr, G., Drummond, T.: Going out: robust model-based tracking for outdoor augmented reality. In: Proceedings of the 5th IEEE and ACM International Symposium on Mixed and Augmented Reality (2006)
22. Rüfenacht, D., Brown, M., Beutel, J., Süsstrunk, S.: Temporally consistent snow cover estimation from noisy, irregularly sampled measurements. In: Proc. 9th International Conference on Computer Vision Theory and Applications (2014)
23. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: Proceedings of the 19th international conference on World wide web. pp. 851–860. ACM (2010)
24. Salvatori, R., Plini, P., Giusto, M., et al.: Snow cover monitoring with images from digital camera systems. Italian Journal of Remote Sensing 43, 137–145 (2011)
25. Schnebele, E., Cervone, G., Waters, N.: Road assessment after flood events using non-authoritative data. Natural Hazards and Earth System Science (2014)
26. Sullivan, B.L., Wood, C.L., Iliff, M.J., Bonney, R.E., Fink, D., Kelling, S.: ebird: A citizen-based bird observation network in the biological sciences. Biological Conservation 142(10), 2282–2292 (2009)
27. Wang, J., Korayem, M., Crandall, D.J.: Observing the natural world with flickr. In: Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on. pp. 452–459. IEEE (2013)
28. Zhang, H., Korayem, M., Crandall, D.J., LeBuhn, G.: Mining photo-sharing websites to study ecological phenomena. In: Proceedings of the 21st international conference on World Wide Web. pp. 749–758. ACM (2012)

29. Zook, M., Graham, M., Shelton, T., Gorman, S.: Volunteered geographic information and crowdsourcing disaster relief: a case study of the haitian earthquake. Available at SSRN 2216649 (2010)