Learning Contours for Automatic Annotations of Mountains Pictures on a Smartphone

Lorenzo Porzi
Fondazione Bruno Kessler
Trento, Italy
University of Perugia
Perugia, Italy

Samuel Rota Buló
Fondazione Bruno Kessler
Trento, Italy

Oswald Lanz
Fondazione Bruno Kessler
Trento, Italy

Paolo Valigi
University of Perugia
Perugia, Italy

Elisa Ricci
Fondazione Bruno Kessler
Trento, Italy
University of Perugia
Perugia, Italy

ABSTRACT
In the last few years the ubiquity and computational power of modern smartphones, together with the significant progresses made on wireless broadband technologies, have made Augmented Reality (AR) technically feasible in consumer devices. In this paper we present an AR application for mobile phones to augment pictures of mountainous landscapes with geo-referenced data (e.g. the peaks’ names, positions of mountain dews or hiking tracks). Our application is based on a novel approach for image-to-world registration, which exploits different information collected with on-board sensors. First, GPS and inertial sensors are used to compute a rough estimate of device position and orientation, then visual cues are exploited to refine it. Specifically, a new learning-based contour detection method based on Random Ferns is used to extract visible mountain profiles from a picture, which are then aligned to synthetic ones obtained from Digital Elevation Models. This solution guarantees an increased accuracy with respect to previous works based only on sensors or on standard edge detection and filtering algorithms. An experimental evaluation conducted on a large set of manually aligned photographs demonstrates that the proposed registration method is both accurate in reconstructing camera position and orientation, and computationally efficient when implemented on a smartphone.

Categories and Subject Descriptors
H.5.1 [Multimedia Information Systems]: Artificial, augmented, and virtual realities

Keywords
Augmented reality, inertial sensors, image annotation.

1. INTRODUCTION
Augmented Reality technologies allow for a digitally enhanced view of the real world. AR applications are many and encompass different areas such as tourism (e.g. superimposing descriptions of monuments on buildings facades), entertainment (e.g. inserting characters into real scenes for gaming), civil engineering and construction (e.g. visualizing technical layouts and infrastructure information) or medicine (e.g. providing assistance to surgeons during operations). Many AR applications rely on publicly available archives of geographic referenced data (e.g. geo-referenced photo collections, geoscience data). For example, in [6] an AR approach is proposed where geo-data are exploited to label mountains in a picture, by reporting their heights above the sea level and their distances from the camera. In [13] an AR system for visualizing geo-located wine-growing data is presented. These systems work well, i.e. labels or markers are accurately and precisely superimposed to the associated objects in the pictures, but they operate “offline” as they typically rely on computationally expensive processes or human feedback. Thus, they are not appropriate to be implemented into mobile devices.

Recently, the large diffusion of smartphones and tablets equipped with on-board GPS, inertial sensors and with increased processing power, has enabled AR systems to run entirely on mobile devices in real-time. However, many current systems only make use of embedded sensors to estimate the device pose and position. This is practically insufficient in most AR applications, which need a precise and solid registration between the real scene and the synthetic representation of the world. In fact, an accurate registration...
is essential to exactly label points of interest into a visual scene.

In this paper we present an AR system for the automatic annotation of pictures of mountainous landscapes, which runs on a mobile phone. Typical examples of augmented contents include notable landmarks, mountain peaks, hiking trails, names of de
ews, positions of other users, etc. Figure 1 provides an example of the output of our system: a picture taken by a smartphone where mountains’ peaks are labeled with their names. Our system relies on a novel 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). Our 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 novel alignment algorithm that exploits visual information. Similarly to [4, 6], our algorithm matches edges extracted from the given image against synthetic profiles, guided by a scoring function supporting the best alignment. However, motivated by the need of devising a computationally efficient approach suitable for mobile devices, we depart from [4, 6] by adopting a very simple and fast scoring function and devise a more sophisticated, learning-based edge detection approach. By doing so, we prevent 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. The user can then explore the augmented picture using the touch gestures commonly employed in smartphone interfaces (e.g. pinch-to-zoom).

A main contribution of this paper, inspired by recent works in computer vision [9, 10], consists in casting the problem of contour detection as a classification problem. This is motivated by our interest in locating mountain profiles instead of detecting edges indiscriminately. Specifically, we propose to use Random Ferns [15] as 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 [5] and Compass [20], 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). This intuition is confirmed by the experimental evaluation conducted on a dataset of thousands of manually annotated pictures, where we show that our algorithm yields accurate registrations, while being fast enough to be implemented on a mobile phone.

This paper is organized as follows. In Sec. 2 we review related works. Section 3 describes the proposed AR system for annotating mountains pictures with peaks’ names. In Sec. 4 the results of our experimental evaluation are provided while in Sec. 5 conclusions are drawn.

2. RELATED WORKS

Several mobile applications exist with a work-flow that is comparable to the one of our system (picture acquisition, offline processing, interaction), although being focused on different tasks. Notably, the Google Camera App shipped with recent versions of the Android mobile OS offers three modes of operation that, similarly to our system, exploit inertial sensors and vision: Photo Sphere [1], Panorama and Lens Blur [2]. In the first two modes, the user, guided by the phone’s sensors, takes several pictures of the environment, which are stitched together after an off-line elaboration phase that takes several seconds. The final result can then be navigated using a touch-based interface. The Lens Blur mode applies a “bokeh” effect to a picture by exploiting a second picture of the same subject, taken from a slightly different angle. After an initial elaboration, in which a depth map of the environment is computed, the user can adjust the position of the virtual focus point and export the result.

The specific problem of superimposing contents to pictures of mountainous landscapes has been considered in some previous works [3, 4, 6]. In [4, 6], picture-to-world registration is performed by aligning profiles obtained from a virtual panorama rendered using DEM information with an edge map computed on the given image. In [3], no initial estimate is assumed for the user position, thus a radically different approach based on bag-of-words is developed. Our approach, instead, is closer to [4, 6], where on-board sensors are employed to roughly estimate the user’s location, and where an image-to-profiles matching based on contours is performed. However, conversely to our work, Baboud et al. [4] focused on devising an effective score function to compare synthetic and detected profiles, while employing a standard edge detection method [20]. Moreover, importantly, none of the previous methods [4, 6] is efficient enough to be run on a mobile device.

Our approach, differently from the previous ones [3, 4, 6], does not rely on filtering or standard edge detection algorithms, for contours pixels are found using a learning-based method. This provides significant benefits both in terms of accuracy and computational costs. Learning-based contour detection has been explored in several previous works [9, 10, 18, 19]. Dollar et al. [9] used a boosting-based approach to independently classify each pixel from its surrounding image patch. More recently [10] achieved improved results by employing a Structured Random Forest classifier, which also considers correlations between adjacent output pixels. Ren et al. [19] achieved state-of-the-art results by employing a method based on sparse coding which, however, suffers from high computational cost. Particularly relevant to our work is the method of Prasad et al. [18], which classifies according to different edge classes the pixels already recognized as edges by a Canny edge detector.

3. SYSTEM DESCRIPTION

Our system takes as input a photo of a mountainous landscape and augments it with geo-referenced information. To do that, it exploits the information provided by the inertial, magnetic and GPS sensors integrated in the smartphone, together with the DEM of the environment. Moreover, it is designed in a way to minimize network communication, thus being as self-contained as possible, for cellular networks are usually not reliable enough in mountainous environments. Figure 2 shows a block diagram of the proposed AR system which is composed of four main blocks: Contour Detection, 3D Rendering, Registration and Augmentation. All modules are implemented on the smartphone, excepting 3D Rendering that runs on a remote server. This approach restricts network communication to a bare minimum: in the 3D Ren-
dering block the server receives a pair of GPS coordinates from the phone and uses the DEM data to calculate the synthetic mountain profiles in a 360° field of view around that location. The profiles are then sent back to the phone, together with a set of relevant geo-referenced items that will be used to augment the photo in the last step of the pipeline.

Initially, the location of the smartphone gathered from the GPS sensor is sent by the device to the remote server, where the 3D Rendering module (see, Sec. 3.2) collects the DEM data and the geo-referenced items. In the meanwhile, the Contour Detection module on the smartphone detects mountain edges in the input image and determines an initial estimate of the viewpoint’s orientation from the inertial and magnetic sensors (see, Sec. 3.1). The final orientation is then computed in the Registration module by optimizing a score function in a neighborhood of the initial estimate (see, Sec. 3.3). The score function compares the detected contours with the synthetic profiles received from the remote server, given a candidate camera orientation. Finally, the Augmentation module superimposes on the image the relevant geo-referenced items, i.e., names and altitudes of mountain peaks retrieved from GeoNames ¹. An example of the system’s final output, i.e., an image where names are superimposed to mountains peaks, is shown in Fig. 1.

3.1 Contour Detection

The Contour Detection block takes as input an image and produces a contour map, which associates every pixel with a probability of being a contour point. In general, an input image will contain many perceptual edges, but only the ones corresponding to discontinuities in the terrain are of interest for our purpose. Standard edge detection techniques are unable to perform this kind of discrimination, thus producing many spurious, irrelevant edges, which often compromise the quality of the subsequent registration step. To fill this semantic gap, we propose to learn by examples which edges are salient for our purpose, thus regarding edge detection as a classification problem. Instead of classifying each pixel of an image as being part of a relevant contour, e.g., as done in [9, 10], we follow an approach similar to [18] by preselecting putative contour edges using an unsupervised, fast edge detector. This results in a computationally efficient solution, for a large portion of pixels will be excluded a priori from the classification, at the expense of a possible loss in the accuracy of the final prediction in the cases when relevant contour pixels are not recognized by the baseline edge detector. This is a price we afford, given the relatively low computational power available on a mobile device. It is worth of notice that none of these aforementioned works [9, 10, 18] has addressed the problem of detecting contours of mountains profiles, their emphasis begin focused on other types of objects and natural scenes.

Random Ferns.

We cast the edge detection problem in terms of a classification problem, in which the input space \( \mathcal{X} \) consists of triplets \( x = (I, u, v) \in \mathcal{X} \) identifying a pixel at coordinates \((u, v)\) in image \(I\), and the output space \( \mathcal{Y} = \{+1, -1\}^{w \times w} \) consists of contour predictions of a \(w \times w\) patch around pixel \((u, v)\). Entries of a patchwise prediction \( Y \in \mathcal{Y} \), denoted by \( y_{h \ell} \), indicate whether the neighbour of \((u, v)\) corresponding to \((h, \ell)\) in the prediction patch is a contour element \((Y_{h \ell} = +1)\), or not \((Y_{h \ell} = -1)\). Moreover, we denote by \(\Psi\) the set of binary features that can be defined on \(\mathcal{X}\). Each binary feature is a function \(\psi : \mathcal{X} \rightarrow \{0, 1\}\) assigning a binary value to samples in \(\mathcal{X}\).

The learning-based edge detector that we propose is based on the Random Ferns (RFN) classifier [16]. Random Ferns can be regarded as a semi-naive Bayes classifier, which combines an ensemble of independent predictors called “ferns”. A random fern is a vector-valued function \(f = (f_1, \ldots, f_m)\), each component being a random, binary feature from \(\Psi\), i.e. \(f_i \in \Psi\) for all \(i = 1, \ldots, k\). A set \(\mathcal{F} = \{f^1, \ldots, f^m\}\) of \(m\) random ferns forms a RFN classifier. Training a RFN classifier \(\mathcal{F}\) amounts to learning for each single fern \(f^j \in \mathcal{F}\) a conditional probability distribution of the fern’s output given a class label, such that we can evaluate for each \(x \in \mathcal{X}\), \(Y \in \mathcal{Y}\) the probability \(P(f^j(x)|Y)\). Given this information, the class posterior probability can be derived as

\[
P(Y|x, \mathcal{F}) \propto \prod_{j=1}^{m} \prod_{h \ell=1}^{w} P(f^j(x)|Y_{h \ell}),
\]

where we implicitly assumed uniform prior over \(\mathcal{Y}\), conditionally independent ferns and independent prediction over the elements of \(Y\).

Given a training set \(\mathcal{T} = \{(x^s, Y^s)\}_{s=1}^{n} \subseteq \mathcal{X} \times \mathcal{Y}\) consisting of \(n\) labelled samples, the conditional probability distribution associated to each fern \(f^j\) can be learned from the data by means of a maximum-a-posteriori estimate under uniform, independent Dirichlet priors with parameter \(\alpha > 0\).

¹http://www.geonames.org
This yields the following learning rule:

\[ P(f^s(x)|y_h) \propto \alpha + \sum_{k=1}^{n} 1_{f^s(x),y_h} \quad \sum_{k=1}^{n} \frac{d(r_k^s)}{C(\text{Proj}(x,p_k^s; k))}, \quad \text{(2)} \]

where 1 is the indicator function giving 1 if the subscript proposition is true, 0 otherwise. The term on the right counts the number of training samples \((x^s,y^s)\) having \(y_h = y_h\) and feature representation \(f^s(x^s) = f^s(x)\).

**Contour map creation.**

Once the RFN classifier has been trained, Equation 1 can be used to build a probabilistic contour map at test time for a novel image \(I\). As mentioned in Sec. 3.1, we exploit a baseline edge detector to pre-select putative contour pixels. Specifically, we apply a simple Sobel filter to image \(I\) to compute an approximate gradient magnitude for every pixel. Each pixel \((u,v)\) exhibiting a magnitude larger than \(\tau\) originates a sample \(x = (u,v,I)\) that is fed to our trained RFN classifier. This yields predictions for all pixels in a \(w \times w\) neighbourhood of \((u,v)\). Note that, by adopting this approach, pixels get multiple predictions due to overlapping neighborhoods, which are averaged in the final map.

**Binary features.**

The type of binary features that we adopt resemble the ones used in [15], with the important difference that we encode a multi-resolution component, which implicitly allows us to inspect pixel values at different scales. In detail, a random, binary feature \(f(x; \theta) \in \Psi\) is a function parametrized by a tuple \(\theta = (\delta u_1, \delta v_1, \delta u_2, \delta v_2, \sigma, c, \rho)\). Given its argument \(x = (u,v,I)\), the parameters \(\delta u_1, \delta v_1\) identify the coordinates of two random neighbours of \((u,v)\) in image \(I\) at coordinates \((u + \delta u_1, v + \delta v_1)\) and \((u + \delta u_2, v + \delta v_2)\). \(\sigma > 0\) is a scale value that indicates at which resolution the value of the parameterization, the binary feature is defined as

\[ f(x; \theta) = \begin{cases} 1 & \text{if } I^c_{\sigma}(u_1, v_1) - I^c_{\sigma}(u_2, v_2) > \rho, \\ 0 & \text{otherwise}, \end{cases} \quad \text{(3)} \]

where we abbreviated the first and second random neighbour by \((u_1, v_1)\) and \((u_2, v_2)\), and denoted by \(I^c_{\sigma}(u,v)\) the intensity value of pixel \((u,v)\) in channel \(c\) at scale \(\sigma\). Note that to obtain the intensity values at different scale there is no need to pre-compute an image scale pyramid. Indeed, those values can be obtained in constant time given the integral image of \(I\) [7]. Finally, the range in which \(\rho\) has to be sampled is determined by the maximum and minimum value of the difference in (3) computed with respect to the samples in the training set.

**3.2 3D Rendering**

The 3D Rendering block exploits publicly available elevation data from the CGIAR-CSI [12] dataset and the Viewfinder Panoramas [8] dataset. A ray-casting algorithm, which also takes into account earth curvature and atmospheric refraction, is used to calculate a 2D cylindrical projection of the environment around the viewpoint. The result is a depth map that associates a distance from the observer to every cylindrical coordinate. The mountains profiles are then extracted from this depth map by calculating the gradient magnitude, applying a threshold and finally performing a simple vectorization procedure on the result.

**3.3 Registration**

The Registration block takes as input the contour map computed by the Contour Detection block, the initial camera orientation estimated from sensor data and a set of profiles received from the server, and produces as output a refined orientation. The mountain profiles are represented as sets \(P = \{P_1, \ldots, P_T\}\), with \(P_i = \{p_i^1, \ldots, p_i^{n_i}\}\). The points \([\theta_1^i, \phi_1^i, r_1^i]\) are expressed in spherical coordinates relative to a world-fixed coordinates frame centered on the user’s GPS coordinates. The camera orientation is parameterized with a triplet \(x = [y p r]\) of yaw, pitch and roll angles. We denote as \(\text{Proj}(x,p_k^s; k)\) the function that, given an orientation \(x\) and the camera intrinsic parameters \(k\), projects the 3D profile point \(p\) on the coordinates \((u,v)\) of the image plane. In this paper we consider a simple pin-hole projection model. Denoting as \(x_S = [y_S p_S r_S]\) the orientation calculated from the smartphone’s sensors, we obtain the final orientation estimate by solving:

\[
\max_x \quad \sum_{i=1}^{T} \sum_{j=1}^{n_i} d(r_k^s) \cdot C(\text{Proj}(x,p_k^s; k)), \quad \text{s.t.}\]

\[
x \geq x_S - b_i, \quad x \leq x_S + b_u,
\]

where \(b_i\) and \(b_u\) define respectively a lower and an upper bound to a search space centered around \(x_S\) and \(d(r_k^s)\) is a function of the distance of the profile point from the observer. The objective of \(d(\cdot)\) is that of giving more importance to distant profiles, as the accuracy of the DEM increases when the distance from the viewpoint increases. In our current implementation \(d(\cdot)\) is a step function that weights 1 the points beyond a predefined distance and 0 the others. However, other weighting schemes are possible as well. We observe that the objective function of (4) is highly non-linear, has many local maxima and is pretty expensive to compute. For these reasons we chose to solve (4) using a stochastic optimization algorithm. In particular, we adopt the Accelerated Particle Swarm Optimization in [21].

**4. EXPERIMENTAL RESULTS**

In our experiments we consider two different datasets: one is used to test the performance of the proposed algorithm, the other for training. As described in Sec.3, the dataset used for training comprises 100 images taken from 100 different locations in the Alps downloaded from Flickr. The test set consists of 12 outdoors sequences recorded from several locations in the Alps. Each sequence has been captured using a Sony Ericsson XPERIA Z smartphone, and comprises between 150 and 500 frames with a resolution of 640 x 480 pixels, for a total of 3117 images, together with a dump of the phone’s sensors and GPS coordinates. We manually aligned all images in both datasets to the DEM to derive their absolute orientation, then, using the algorithms described in Sec. 3.2, we projected a set of synthetic mountain profiles onto each image plane. Finally, for the images in the training set, we labeled each pixel lying on a profile as “contour” and every other pixel as “not-contour”.

In the next paragraphs, if not otherwise noted, we consider the following fixed parameter values for RFN: \(r = 8, k = 12\).
In the first series of experiments we evaluate the performance of the proposed system, analyzing the impact of different contour detection algorithms on both registration accuracy and computational times. Specifically, we consider the proposed RFN approach analyzing the influence of the threshold \( \tau \) and the number of ferns \( m \) while fixing \( w = 4 \). We compare RFN with a standard edge detector (i.e., Canny [5]) that was also used for registering mountains pictures in previous works [6]. As a baseline we also show the results obtained using sensor-based orientation estimation, i.e. without exploiting visual cues. Figure 3 compares the different approaches showing the average registration error (Fig. 3, left) and the computational times (Fig. 3, right) when they run on a Sony XPERIA Z smartphone. As expected we achieve close to real-time performance when using only the sensors or the Canny edge detector, but the resulting registration accuracy is generally insufficient for an AR application (average error greater than \( 5^\circ \)). Our RFN approach, on the other side, guarantees a significantly lower registration error (\( \sim 1.3^\circ \) for some choice of parameters). In this case the associated computational time is in the order of a few seconds, which is reasonable for our application. From the results shown in Fig. 3 we can also observe the influence of different parameters: \( m = 100 \) ferns and a threshold \( \tau = 0.2 \) seem to be the values of choice according to a trade-off between registration error and running time (small values of \( \tau \) and large values of \( m \) increase the running time). The influence of the Sobel threshold \( \tau \) and the output patch size \( w \) is further investigated in Fig. 6, which shows the registration error for our RFN approach when \( m = 100 \). Generally, increasing \( \tau \) the registration accuracy degrades as expected. Setting \( w = 4 \) or \( w = 8 \) guarantees superior accuracy with respect to \( w = 1 \), as a wider output patch leads to a smoother and less noisy edge map.

In a second series of experiments we compare two learning-based and two standard contour detectors. In particular we consider: the proposed RFN, the Random Forest classifier with structured output in [10], the Canny [5] edge detector and the Compass [20] edge detector. Both Canny and Compass have been adopted in previous works [6, 4] for registering pictures of mountainous terrains. Table 1 shows the mean registration error for the 12 test sequences when using these different approaches. The advantage of employing a learning-based contour detection algorithm is evident, as Canny and Compass are consistently outperformed by our RFN method and by the Random Forest classifier. In most sequences the Random Forest classifier gives the best results: this is somehow expected as its output is generally smoother and less noisy compared to the other contour detectors. However, this approach and in general any structured output classifier typically relies on a complex inference algorithm which is too computationally expensive to run on a mobile device. Similarly, the Compass detector produces a more accurate edge map than Canny, but its high computation complexity makes it unsuitable for our application.

It is worth noting that the estimation error for the proposed methods varies significantly among different sequences. For instance, the first two images of Fig. 4 correspond to the system output on the first frame of sequence 12 when Canny and RFN are used as contour detectors. In this case we observe a small alignment error for both detectors (resp. \( \sim 0.9^\circ \) and \( \sim 0.3^\circ \)), which translates to an accurate augmentation. The last two images of Fig. 4 show the system output for the first frame of sequence 2 and the same contour detectors. Here RFN produces an error of \( \sim 4^\circ \) resulting in an inaccurate but still intelligible augmentation, while Canny produces an error of \( \sim 16^\circ \) which is not acceptable for our application. In general RFN produces significantly better results than traditional approaches such as Canny or Compass in all those cases where the scene is highly cluttered or presents spurious edges. Our approach, in fact, is able to discard many edges that are not due to mountain profiles, as it is evident from Fig. 5.

5. CONCLUSIONS

We presented an AR application for annotating mountain pictures running on a mobile phone. Our system is based on a novel approach for photo-to-world registration which jointly exploits information provided by GPS, inertial sensors and visual cues. Our experiments on a large dataset of manually annotated photographs clearly demonstrate that our registration method guarantees robust orientation estimates and thus precise content augmentation. Future works will explore other approaches based on structured output learning for contour detection [14], novel optimization solutions for improving computational efficiency and how to integrate this algorithm into an ego-motion estimation framework [17] selecting adaptively discriminative features [11].

![Figure 3: Comparison of different approaches according to (left) registration error and (right) running time.](image_url)
Figure 4: System output for the first frames of sequences 1 and 2 when using RFn(A,D) and Canny (B,C).

Figure 5: First frame of sequence 5 (A) and contour maps obtained with the Canny (B), Compass (C) and our approach (D).

Figure 6: Registration errors obtained with our approach at varying values of the patch size $w$ and the threshold $\tau$.

6. ACKNOWLEDGMENTS
This research has been partly funded by the European 7th Framework Program, under grant VENTURI (FP7-288238).

7. REFERENCES