Noname manuscript No.
(will be inserted by the editor)

Image Based Geo-Localization in the Alps

Olivier Saurer · Georges Baatz · Kevin Köser · L'ubor Ladický · Marc Pollefeys

Received: date / Accepted: date

Abstract Given a picture taken somewhere in the world, automatic geo-localization of such an image is an extremely useful task especially for historical and forensic sciences, documentation purposes, organization of the world's photographs and intelligence applications. While tremendous progress has been made over the last years in visual location recognition within a single city, localization in natural environments is much more difficult, since vegetation, illumination, seasonal changes make appearance-only approaches impractical. In this work, we target mountainous terrain and use digital elevation models to extract representations for fast visual database lookup. We propose an automated approach for very large scale visual localization that can efficiently exploit visual information (contours) and geometric constraints (consistent orientation) at the same time. We validate the system at the scale of Switzerland (40 000km^2) using over 1000 landscape query images with ground truth GPS position.

Keywords Geo-Localization · Localization · Camera Calibration · Computer Vision

1 Introduction and Previous Work

In intelligence and forensic scenarios as well as for searching archives and organising photo collections, automatic image-based location recognition is a challenging

Olivier Saurer
ETH Zürich, Switzerland, E-mail: saurero@inf.ethz.ch

Georges Baatz
Google Inc., Zürich, Switzerland, E-mail: gbaatz@google.com

Kevin Köser
GEOMAR Helmholtz Centre for Ocean Research Kiel, Germany, E-mail: kkoeser@geomar.de

L'ubor Ladický
ETH Zürich, Switzerland, E-mail: lubor.ladicky@inf.ethz.ch

Marc Pollefeys
ETH Zürich, Switzerland, E-mail: marc.pollefeys@inf.ethz.ch

task that would be extremely useful when solved. In such applications GPS tags are typically not available in the images requiring a fully image-based approach for geo-localization. Over the last years progress has been made in urban scenarios, in particular with stable man-made structures that persist over time. However, recognizing the camera location in natural environments is substantially more challenging, since vegetation changes rapidly during seasons, and lighting and weather conditions (e.g. snow lines) make the use of appearance-based techniques (e.g., patch-based local image features [28, 8]) very difficult. Additionally, dense street-level imagery is limited to cities and major roads, and for mountains or for the countryside only aerial footage exists, which is much harder to relate with terrestrial imagery.

In this work we give a more in depth discussion on camera geo-localization in natural environments. In particular we focus on recognizing the skyline in a query image, given a digital elevation model (DEM) of a country — or ultimately, the world. In contrast to previous work of matching e.g. a peak in the image to a set of mountains known to be nearby, we aggregate shape information across the whole skyline (not only the peaks) and search for a similar configuration of basic shapes in a large scale database that is organized to allow for query images of largely different fields of view. The method is based on sky segmentation, either automatic or easily supported by an operator for challenging pictures such as those with reflection, occlusion or taken from inside a cable car.

Contributions.

A preliminary version of this system was presented in [2]. This work provides a more detailed analysis and evaluation of the system and improves upon the skyline segmentation. The main contributions are a novel method for robust contour encoding as well as two different voting schemes to solve the large scale camera pose recognition from contours. The first scheme operates only in descriptor space (it verifies where in the model a panoramic skyline is most likely to *contain* the current query picture) while the second one is a combined vote in descriptor and rotation space. We validate the whole approach using a public digital elevation model of Switzerland that covers more than 40 000km^2 and a set of over 1000 images with ground truth GPS position. In particular we show the improvements of all novel contributions compared to a baseline implementation motivated by classical bag-of-words [31] based techniques like [8]. In addition we proposed a semi-automatic skyline segmentation technique, based on a dynamic programming approach. Furthermore, we demonstrate that the skyline is highly informative and can be used effectively for localization.

Previous Work.

To the best of our knowledge this is the first attempt to localize photographs of natural environments at large scale based on a digital elevation model. The closest works to ours are smaller scale navigation and localization in robotics [37, 32], and building/location recognition in cities [28, 1, 8, 26, 34, 4] or with respect to community photo collections of popular landmarks [19]. These, however, do not apply to landscape scenes of changing weather, vegetation, snowlines, or lighting conditions.

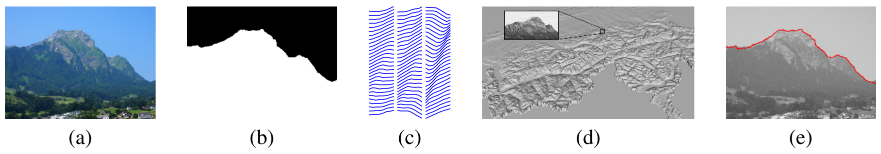


Fig. 1 Different stages in the proposed pipeline: (a) Query image somewhere in Switzerland, (b) sky segmentation, (c) sample set of extracted 10° contourlets, (d) recognized geo-location in digital elevation model, (e) overlaid skyline at retrieved position.

The robotics community has considered the problem of robot navigation and robot localization using digital elevation models for quite some time. Talluri et al. [33] reason about intersection of known viewing ray directions (north, east, south, west) with the skyline and relies thus on the availability of 360° panoramic query contours and the knowledge of vehicle orientation (i.e. north direction). Thompson et al. [35] suggest general concepts of how to estimate pose and propose a hypothesize and verification scheme. They also rely on known view orientation and match viewpoint-independent features (peaks, saddle points, etc.) of a DEM to features found in the query image, ignoring most of the signal encoded in the skyline. In [11], computer vision techniques are used to extract mountain peaks which are matched to a database of nearby mountains to support a remote operator in navigation. However, we believe that their approach of considering relative positions of absolute peaks detected in a DEM is too restrictive and would not scale to our orders of magnitude larger problem, in particular with respect to less discriminative locations. Naval et al. [24] proposes to first match three features of a contour to a DEM and estimate an initial pose from that before doing a non-linear refinement. Also here the initial step of finding three correct correspondences is a challenging task in a larger scale database. Stein et al. [32] assumes panoramic query data with known heading, and computes super-segments on a polygon fit, however descriptiveness/robustness is not evaluated on a bigger scale, while [10] introduces a probabilistic formulation for a similar setting. The key point is that going from tens of potential locations to millions of locations requires a conceptually different approach, since exhaustive image comparison or trying all possible "mountain peaks" simply does not scale up to a large-scale geo-localization problems. Similarly, for urban localization, in [27] an upward looking 180° field-of-view fisheye is used for navigation in urban canyons. They render untextured city models near the predicted pose and extract contours for comparison with the query image. A similar approach was recently proposed by Taneja et al. [34], where panoramic images are aligned to a cadastral 3D model by maximizing the overlap between the panoramic image and the rendered model. In [26] Ramalingam et al. propose a general framework to solve for the camera pose using 3D-to-2D point and line correspondences between the 3D model and the query image. The approach requires an initial correspondence match, which is propagated to the next image using appearance based matching techniques. These approaches are meant as local methods for navigation or pose refinement. Also recently, in [3] Baboud et al. optimize the camera orientation given the exact position, i.e. they estimate the viewing direction given a good GPS tag. In [4] Bansal et al. propose a novel correspondence-free geo-localization approach in urban environments. They match corners and roof-line edges of buildings to a database of 3D corners and direction vectors previously extracted from a DEM. None of the above mentioned systems considered recognition and localization in natural environments at large scale.

On the earth scale, Hays et al. [13] source photo collections and aim at learning location probability based on color, texture, and other image-based statistics. Conceptually, this is not meant to find an exact pose based on geometric considerations but rather discriminates landscapes or cities with different (appearance) characteristics on a global scale. In [18] Lalonde et al. exploit the position of the sun (given the time) for geo-localization. In the same work it is also shown that identifying a large piece of clear sky without haze provides information about the camera pose (although impressive given the data, over 100km mean localization error is reported). Both approaches are appealing for excluding large parts of the earth from further search but do not aim at exactly localizing the camera within a few hundred meters.

Besides attacking the DEM-based, large scale geo-localization problem we propose new techniques that might also be transferred to bag-of-words approaches based on local image patches (e.g. [31,28,8]). Those approaches typically rely on pure occurrence-based statistics (visual word histogram) to generate a first list of hypotheses and only for the top candidates geometric consistency of matches is verified. Such a strategy fails in cases where pure feature coocurrence is not discriminative but where the relative locations of the features are important. Here, we propose to do a (weak) geometric verification already in the histogram distance phase. Furthermore, we show also a representation that tolerates largely different document sizes (allowing to compare a panorama in the database to an image with an order of magnitude smaller field-of-view).

2 Mountain Recognition Approach

The location recognition problem in its general form is a six-dimensional problem, since three position and three orientation parameters need to be estimated. We make the assumption that the photographs are taken not too far off the ground and use the fact that people rarely twist the camera relative to the horizon [7] (e.g. small roll). We propose a method to solve that problem using the outlines of mountains against the sky (i.e. the skyline). For the visual database we seek a representation that is robust with respect to tilt of the camera which means that we are effectively left with estimating the 2D position (latitude and longitude) on the digital elevation model and the viewing direction of the camera. The visible skyline of the DEM is extracted offline at regular grid positions (360° at each position) and represented by a collection of vector-quantized local contourlets (contour words, similar in spirit to visual words obtained from quantized image patch descriptors [31]). In contrast to visual word based approaches, additionally an individual viewing angle α_d ($\alpha_d \in [0; 2\pi]$) relative to north direction is stored. At query time, a skyline segmentation technique is applied that copes with the often present haze and also allows for user interaction in case of incorrect segmentation. Subsequently the extracted contour is robustly described by a set of local contourlets plus their relative angular distance α_q with respect to the optical axis of the camera. The contour words are represented as an inverted file system, which is used to query the most promising location. At the same time the inverted file also votes for the viewing direction, which is a geometric verification integrated in the bag-of-words search.

2.1 Processing the Query Image

2.1.1 Sky Segmentation

The estimation of the visible skyline can be cast as a foreground-background segmentation problem. As we assume almost no camera roll and since overhanging structures are not modelled by the 2.5D DEM, finding the highest foreground pixel (foreground height) for each image column provides an good approximation and allows for a dynamic programming solution, as proposed in [20,5]. To obtain the data term for a candidate height in a column we sum all foreground costs below the candidate contour and all sky costs above the contour. The assumption is, when traversing the skyline, there should be a local evidence in terms of an orthogonal gradient (similar in spirit to flux maximization [36] or contrast sensitive smoothness assumptions [6, 15] in general 2D segmentation).

We express the segmentation problem in terms of an energy:

$$E = \sum_{x=1}^{width} E_d(x) + \lambda \sum_{x=1}^{width-1} E_s(x, x+1), \tag{1}$$

where E_d represents the data term, E_s the smoothness term and λ is a weighting factor. The data term $E_d(x)$ in one column x evaluates the cost of all pixel below it to be assigned a foreground label while all pixels above it are assigned a background (sky) label. The cost is incorporated into the optimization framework as a standard negative-log-likelihood:

$$E_d = \sum_{i=1}^{k-1} -\log h(\mathscr{F}|z_i) + \sum_{i=k}^{height} -\log h(\mathscr{B}|z_i), \tag{2}$$

where $h(\mathscr{F}|z_i)$ denotes the probability of pixel z_i being assigned to the foreground $\mathscr{F}$ model and $h(\mathscr{B}|z_i)$ the probability of a pixel being assigned to the background $\mathscr{B}$ model. The likelihoods $h(z|\mathscr{F})$ and $h(z|\mathscr{B})$ are computed by the pixel-wise classifier, jointly trained using contextual and superpixel based feature representations [17].

The contextual part of the feature vector [30,16] consists of a concatenation of bag-of-words representations over a fixed random set of 200 rectangles, placed relative to the corresponding pixel. These bag-of-words representations are built using 4 dense features - textons [22], local ternary patterns [14], self-similarity [29] and dense SIFT [21], each one quantized to 512 clusters using standard K-means clustering. For each pixel the superpixel part of the feature vector is the concatenation of a bag-of-words representations of a corresponding superpixel [17] from each unsupervised segmentation. Four superpixel segmentations are obtained by varying the parameters of the MeanShift algorithm [9], see Fig. 2. Pixels, belonging to the same

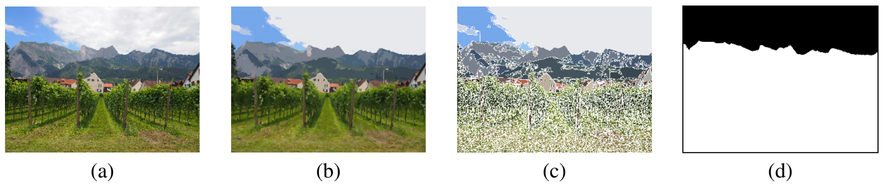


Fig. 2 Superpixel based segmentation: (a) Input image. (b) MeanShift filtered image. (c) MeanShift region boundaries. (d) Final segmentation.

segment, share a large part of the feature vector, and thus tend to have the same labels, leading to segmentations, that follow semantic boundaries.

The most discriminative weak features are found using AdaBoost [12]. The contextual feature representations are evaluated on the fly using integral images [30], the superpixel part is evaluated once and kept in memory. The classifier is trained independently for 5 colour spaces - Lab, Luv, Grey, Opponent and Rgb. The final likelihood is calculated as an average of these 5 classfiers.

The pairwise smoothness term is formulated as:

$$E_s(x, x+1) = \sum_{i \in C} \exp\left(\frac{-\mathbf{d}^\top \mathbf{R} \mathbf{g_i}}{\lambda ||\mathbf{d}||}\right), \tag{3}$$

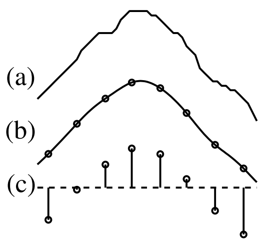
where C is the set of pixels connecting pixel z_n in column x and z_m in column $x+1$ along the Manhattan path (path along the horizontal and vertical direction), $\mathbf{d}$ is the direct connection vector between z_n and z_m, g_i is the image gradient at pixel i, $\mathbf{R}$ represents a 90 degree rotation matrix and λ is set to the mean of $\mathbf{d}^\top \mathbf{R} \mathbf{g_i}$ for each image. The intuition is, that all pixels on the contour should have a gradient orthogonal to the skyline.

Given the energy terms defined in Eq. (2) and (3), the segmentation is obtained by minimizing Eq. (1) using dynamic programming. Our framework also allows for user interaction, where simple strokes can mark foreground or background (sky) in the query image. In case of a foreground labelling this forces all pixel below the stroke to be labels as foreground and in case of a backround stroke, the stroke pixel and all pixels above it are marked as background (sky). This provides a simple and effective means to correct for very challenging situations, where buildings and trees partially occlude the skyline.

2.1.2 Contourlet Extraction

In the field of shape recognition, there are many shape description techniques that deal with closed contours, e.g. [23]. However, recognition based on partial contours is still a largely unsolved problem, because it is difficult to find representations invariant to viewpoint. For the sake of robustness to occlusion, to noise and systematic errors (inaccurate focal length estimate or tilt angle), we decided to use local representations of the skyline (see [38] for an overview on shape features).

To describe the contour, we consider overlapping curvelets of width w (imagine a sliding window, see Fig. 1). These curvelets are then sampled at n equally spaced



(d)

-0.14	-0.01	0.10	0.17	0.15	0.04	-0.10	-0.21

(e)

001	011	101	111	110	100	001	000

(f) 001011101111110100001000

Fig. 3 Contour word computation: (a) raw contour, (b) smoothed contour with n sampled points, (c) sampled points after normalization, (d) contourlet as numeric vector, (e) each dimension quantized to 3 bits, (f) contour word as 24-bit integer.

points, yielding each an n-dimensional vector $\tilde{y}_1, \ldots, \tilde{y}_n$ (before sampling, we low-pass filter the skyline to avoid aliasing). The final descriptor is obtained by subtracting the mean and dividing by the feature width (see Fig. 3(a)–(d)):

$$y_i = \frac{\tilde{y}_i - \bar{y}}{w} \text{ for } i = 1, \ldots, n \qquad \text{where} \qquad \bar{y} = \frac{1}{n} \sum_{j=1}^{n} \tilde{y}_j \tag{4}$$

Mean subtraction makes the descriptor invariant w.r.t. vertical image location (and therefore robust against camera tilt). Scaling ensures that the y_i's have roughly the same magnitude, independently of the feature width w.

In a next step, each dimension of a contourlet is quantized (Fig. 3(e)–(f)). Since the features are very low-dimensional compared to traditional patch-based feature descriptors like SIFT [21], we choose not to use a vocabulary tree. Instead, we directly quantize each dimension of the descriptor *separately*, which is both faster and more memory-efficient compared to a traditional vocabulary tree. In addition the best bin is guaranteed to be found. Each y_i falls into one bin and the n associated bin numbers are concatenated into a single integer, which we refer to as *contour word*. For each descriptor, the viewing direction α_q, relative to the camera's optical axis is computed using the camera's intrinsics parameters and is stored together with the visual word. We have verified that an approximate focal length estimate is sufficient. In case of an unknown focal length, it is possible to sample several tentative focal length values, which we evaluate in Section 3.

2.2 Visual Database Creation

The digital elevation model we use for validation is available from the Swiss Federal Office of Topography, and similar datasets exist also for the US and other countries. There is one sample point per 2 square meters and the height quality varies from 0.5m (flat regions) to 3m-8m (above 2000m elevation) average error[1]. This data is converted to a triangulated surface model with level-of-detail support in a scene graph representation[2]. At each position on a regular grid on the surface (every 0.001° in

[1]`http://www.swisstopo.admin.ch/internet/swisstopo/en/home`
[2]`http://openscenegraph.org`

N-S direction and 0.0015° in E-W direction, i.e. 111m and 115m respectively) and from 1.80m above the ground[3], we render a cube-map of the textureless DEM (face resolution 1024×1024) and extract the visible skyline by checking for the rendered sky color. Overall, we generate 3.5 million cubemaps. Similar to the query image, we extract contourlets, but this time with *absolute* viewing direction. We organize the contourlets in an index to allow for fast retrieval. In image search, inverted files have been used very successfully for this task [31]. We extend this idea by also taking into account the viewing direction, so that we can perform rough geometric verification on-the-fly. For each word we maintain a list that stores for every occurrence the panorama ID and the azimuth α_d of the contourlet.

2.3 Recognition and Verification

2.3.1 Baseline

The baseline for comparison is an approach borrowed from patch based systems (e.g. [25,28,8]) based on the (potentially weighted) L1-norm between normalized visual word frequency vectors:

$$D^E(\tilde{\mathbf{q}},\tilde{\mathbf{d}}) = \|\tilde{\mathbf{q}}-\tilde{\mathbf{d}}\|_1 = \sum_i |\tilde{q}_i - \tilde{d}_i| \quad \text{or} \quad D^{E_w}(\tilde{\mathbf{q}},\tilde{\mathbf{d}}) = \sum_i w_i |\tilde{q}_i - \tilde{d}_i| \tag{5}$$

$$\text{with} \quad \tilde{\mathbf{q}} = \frac{\mathbf{q}}{\|\mathbf{q}\|_1} \quad \text{and} \quad \tilde{\mathbf{d}} = \frac{\mathbf{d}}{\|\mathbf{d}\|_1} \tag{6}$$

Where q_i and d_i is the number of times visual word i appears in the query or database image respectively, and $\tilde{q}_i$, $\tilde{d}_i$ are their normalized counterparts. w_i is the weight of visual word i (e.g. as obtained by the term frequency - inverse document frequency (tf-idf) scheme). This gives an ideal score of 0 when both images contain the same visual words at the same proportions, which means that the L1-norm favors images that are *equal* to the query.

Nister et al. [25] suggested transforming the weighted L1-norm like this

$$D^{E_w}(\tilde{\mathbf{q}},\tilde{\mathbf{d}}) = \sum_i w_i \tilde{q}_i + \sum_i w_i \tilde{d}_i - 2 \sum_{i \in Q} w_i \min(\tilde{q}_i, \tilde{d}_i) \tag{7}$$

in order to enable an efficient method for evaluating it by iterating only over the visual words present in the query image and updating only the scores of database images containing the given visual word.

2.3.2 "Contains"-Semantics

In our setting, we are comparing 10°–70° views to 360° panoramas, which means that we are facing a $5\times$–$36\times$ difference of magnitude. Therefore, it seems ill-advised to

[3]Synthetic experiments verified that taking the photo from ten or fifty meters above the ground does not degrade recognition besides very special cases like standing very close to a small wall.

implement an "equals"-semantics, but rather one should use a "contains"-semantics. We modify the weighted L1-norm as follows:

$$D^C(\mathbf{q},\mathbf{d}) = \sum_i w_i \max(q_i - d_i, 0) \tag{8}$$

The difference is that we are using the raw contour word frequencies, q_i and d_i without scaling and we replace the absolute value $|\cdot|$ by $\max(\cdot,0)$. Therefore, one only penalizes contour words that occur in the query image, but not in the database image (or more often in the query image than in the database image). An ideal score of 0 is obtained by a database image that contains every contour word at least as often as the query image, plus any number of other contour words. If the proposed score is transformed as follows, it can be evaluated just as efficiently as the baseline:

$$D^C(\mathbf{q},\mathbf{d}) = \sum_{i\in Q} w_i q_i - \sum_{i\in Q} w_i \min(q_i, d_i) \tag{9}$$

This subtle change makes a huge difference, see Fig. 6(a) and Table 1: (B) versus (C). Note that this might also be applicable to other cases where a "contains"-semantics is desirable.

2.3.3 Location and Direction

We further refine retrieval by taking geometric information into account already during the voting stage. Earlier bag-of-words approaches accumulate evidence purely based on the frequency of visual words. Voting usually returns a short-list of the top n candidates, which are reranked using geometric verification (typically using the number of geometric inliers). For performance reasons, n has to be chosen relatively small (e.g. $n = 50$). If the correct answer already fails to be in this short-list, then no amount of reordering can bring it back. Instead, we check for geometric consistency already at the voting stage, so that fewer good candidates get lost prematurely. Not only does this increase the quality of the short-list, it also provides an estimated viewing direction, which can be used as an initial guess for the full geometric verification. Since this enables a significant speedup, we can afford to use a longer short-list, which further reduces the risk of missing the correct answer.

If the same contour word appears in the database image at angle α_d (relative to north) and in the query image at angle α_q (relative to the camera's optical axis), the camera's azimuth can be calculated as $\alpha = \alpha_d - \alpha_q$. Weighted votes are accumulated using soft binning and the most promising viewing direction(s) are passed on to full geometric verification. This way, panoramas containing the contour words in the right order get many votes for a single direction, ensuring a high score. For panoramas containing only the right mix of contour words, but in random order, the votes are divided among many different directions, so that none of them gets a good score (see Fig. 4). Note that this is different from merely dividing the panoramas into smaller sections and voting for these sections: Our approach effectively requires that the order of contour words in the panorama matches the order in the query image. As an additional benefit, we do not need to build the inverted file for any specific field-of-view of the query image.

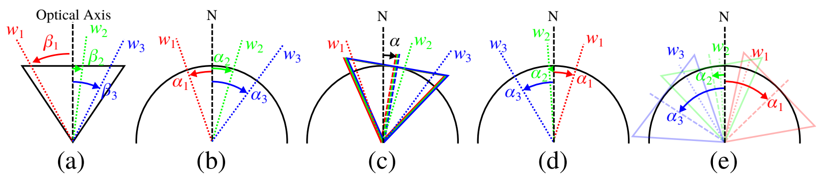


Fig. 4 Voting for a direction is illustrated using a simple example: We have a query image (a) with contour words w_i and associated angles β_i *relative* to the optical axis. We consider a panorama (b) with contour words in the same *relative* orientation α_i as the query image. Since the contour words appear in the same order, they all vote for the same viewing direction α (c). In contrast, we consider a second panorama (d) with contour words in a different order. Even though the contour words occur in close proximity they each vote for a different direction α_i, so that none of the directions gets a high score (e).

2.3.4 Geometric Verification

After retrieval we geometrically verify the top 1000 candidates. The verification consists in computing an optimal alignment of the two visible skylines using iterative closest points (ICP). While we consider in the voting stage only one angle (azimuth), ICP determines a full 3D rotation. First, we sample all possible values for azimuth and keep the two other angles at zero. The most promising one is used as initialization for ICP. In the variants that already vote for a direction, we try only a few values around the highest ranked ones. The average alignment error is used as a score for re-ranking the candidates.

3 Evaluation

In this section we evaluate the proposed algorithm on two real datasets consisting of a total of 1151 images. We further give a detailed evaluation of the algorithm under varying tilt and roll angles, and show that in cases where the focal length parameter is unknown it can effectively be sampled.

Query Set.

In order to evaluate the approaches we assembled two datasets, which we refer to as CH1 and CH2. The CH1 dataset consists of 203 photographs obtained from different sources such as online photo collections and on site image capturing. The CH2 dataset consists of 948 images which were solely captured on site. For all of the photographs, we verified the GPS tag or location estimate by comparing the skyline to the surface model. For the majority of the images the information was consistent. For a few of them the position did not match the digital elevation model's view. This can be explained by a wrong cell phone GPS tag, due to bad/no GPS reception at the time the image was captured. For those cases, we use dense geometric verification (on each 111m$\times$115m grid position up to a 10km radius around the tagged position) to generate hypotheses for the correct GPS tag. We verify this by visual inspection and removed images in case of disagreement. The complete set of query images used

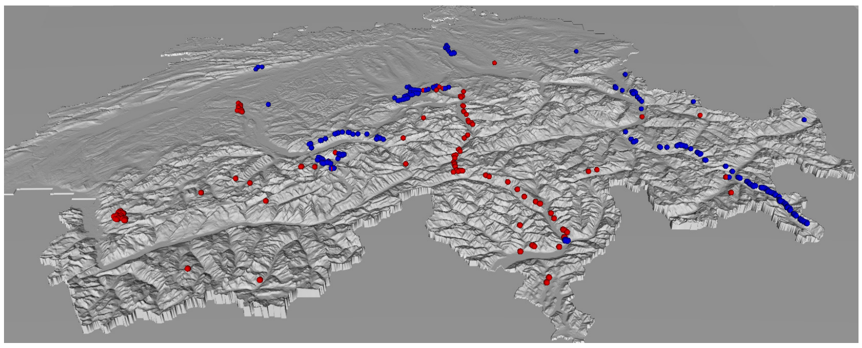


Fig. 5 Oblique view of Switzerland, spanning a total 40 000km^2. Spheres indicate the query images' of the *CH1* (red) and *CH2* (blue) dataset at ground truth coordinates (size reflects 1km tolerance radius). Source of DEM: Bundesamt für Landestopografie swisstopo (Art. 30 GeoIV): 5704 000 000

is available at the project website[4]. The distribution of the CH1 and CH2 dataset is drawn on to the DEM in Fig. 5. For all of the query images FoV information is available (e.g. from EXIF tag). However, we have verified experimentally that also in case of fully unknown focal length the system can be applied by sampling over this parameter, see Fig. 10 as example and subsection 3.

Query Image Segmentation.

We used the CH1 query images which were already segmented in [2] as training set and apply our segmentation pipeline to the CH2 dataset. Out of the 948 image 60% of the images were segmented fully automatically, while 30% required little user interaction, mainly to correct for occluders such as trees or buildings. 10% of the images required a more elaborate user interaction, to correct for snow fields, (often confused as sky), clouds hiding small parts of the mountain or for reflections appearing when taking pictures from inside a car, cable-car or train. Our new segmentation pipeline improved by 18%, compared to the previous method proposed in [2].

Parameter Selection.

The features need to be clearly smaller than the images' field-of-view, but wide enough to capture the geometry rather than just discretization noise. We consider descriptors of width $w = 10°$ and $w = 2.5°$. The number of sample points n should not be so small that it is uninformative (e.g. $n = 3$ would only distinguish concave/convex), but not much bigger than that otherwise it risks being overly specific, so we choose $n = 8$. The curve is smoothed by a Gaussian with $\sigma = \frac{w}{2n}$, i.e. half the distance between consecutive sample points. Descriptors are extracted every σ degrees.

Each dimension of the descriptor is quantized into k bins of width 0.375, the first and last bin extending to infinity. We chose k as a power of 2 that results in roughly 1 million contour words, i.e. $k = 8$. This maps each y_i to 3 bits, producing contour

[4] `http://cvg.ethz.ch/research/mountain-localization`

words that are 24 bit integers. Out of the 2^{24} potential contour words, only 300k–500k (depending on w) remain after discarding words that occur too often (more than a million) or not at all.

Recognition Performance.

The recognition pipeline using different voting schemes and varying descriptor sizes is evaluated on both datasets, see Table 1. All of the tested recognition pipelines return a ranked list of candidates. We evaluate them as follows: For every $n = 1, \dots, 100$, we count the fraction of query images that have at least one correct answer among the top n candidates. We consider an answer correct if it is within 1km of the ground truth position (see Fig. 6).

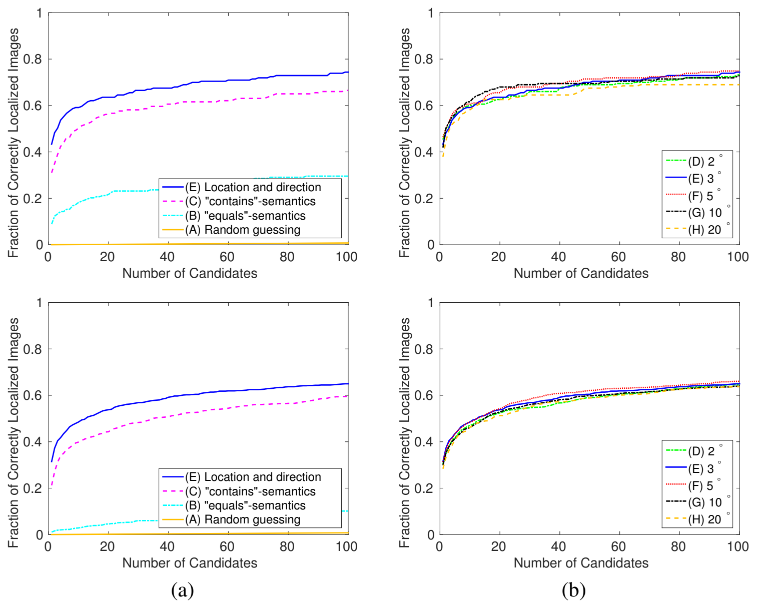


Fig. 6 Retrieval performance for different: (a) voting schemes, (b) bin sizes in direction voting. Evaluated on the CH1 (top) and CH2 (bottom) dataset.

In Fig. 6(a), we compare different voting schemes: (B) voting for location only, using the traditional approach with normalized visual word vectors and L1-norm (“equals”-semantics); (C) voting for location only, with our proposed metric (“contains”-semantics); (E) voting for location and direction simultaneously (i.e. taking order into account). All variants use 10° descriptors. For comparison, we also show (A) the probability of hitting a correct panorama by random guessing (the probability of a correct guess is extremely small, which shows that the tolerance of 1km is not overly generous). Our proposed “contains”-semantics alone already outperforms the baseline (“equals”-semantics) by far, but voting for a direction is even better.

	Voting scheme	Descriptor width	Dir. bin size	Geo. ver.	CH1 (top 1 corr.)	CH2 (top 1 corr.)
(A)	random	N/A	N/A	no	0.008%	0.008%
(B)	"equals"	10°	N/A	no	9%	1%
(C)	"contains"	10°	N/A	no	31%	21%
(D)	loc.&dir.	10°	2°	no	45%	30%
(E)	loc.&dir.	10°	3°	no	43%	31%
(F)	loc.&dir.	10°	5°	no	46%	31%
(G)	loc.&dir.	10°	10°	no	42%	30%
(H)	loc.&dir.	10°	20°	no	38%	28%
(I)	loc.&dir.	2.5°	3°	no	28%	14%
(J)	loc.&dir.	10°&2.5°	3°	no	62%	44%
(K)	loc.&dir.	10°&2.5°	3°	yes	88%	76%

Table 1 Overview of tested recognition pipelines.

In Fig. 6(b), we analyse how different bin sizes for direction voting affects results. (D)–(H) correspond to bin sizes of $2°, 3°, 5°, 10°, 20°$ respectively. While there are small differences, none of the settings outperforms all others consistently: Our method is quite insensitive over a large range of this parameter.

In Fig. 7(a), we study the impact of different descriptor sizes: (E) only 10° descriptors; (I) only 2.5° descriptors; (J) both 10° and 2.5° descriptors combined. All variants vote for location and direction simultaneously. While 10° descriptors outperforms 2.5° descriptors, the combination of both is better than either descriptor size alone. This demonstrates that different scales capture different information, which complement each other.

In Fig. 7(b), we show the effect of geometric verification by aligning the full countours using ICP: (J) 10° and 2.5° descriptors voting for location and direction, without verification; (K) same as (J) but with geometric verification. We see that ICP

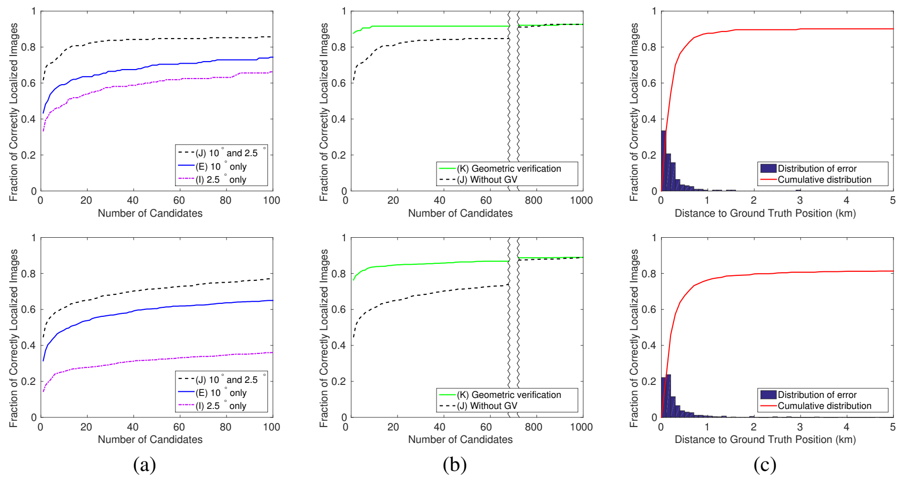


Fig. 7 Retrieval performance for CH1 (top) and CH2 (bottom) dataset: (a) Different descriptor sizes. (b) Retrieval performance before and after geometric verification. (c) Fraction of queries having at most a given distance to the ground truth position. Not shown: 21 images (9.9%) from the CH1 dataset with an error between 7 and 217km and 177 images (18.6%) from the CH2 dataset with an error between 13 and 245km.

based reranking is quite effective at moving the best candidate(s) to the beginning of the short list: On the CH1 dataset the top ranked candidate is within a radius of 1km with a probability of 88%. On the CH2 dataset we achieve a recognition rate of 76% for a maximum radius of 1km. See Fig. 7(c) for other radii. In computer assisted search scenarios, an operator would choose an image from a small list which would further increase the percentage of correctly recovered pictures. Besides that, from geometric verification we not only obtain an estimate for the viewing direction but the full camera orientation which can be used for augmented reality. Fig. 8 and 9 show images of successful and unsuccessful localization.

Field-of-View.

In Fig. 10 we illustrate the effect of inaccurate or unknown field-of-view (FoV). For one query image, we run the localization pipeline (K) assuming that the FoV is 11° and record the results. Then we run it again assuming that the FoV is 12° etc., up to 70°. Fig. 10 shows how the alignment error and estimated position depend on the assumed FoV.

In principle, it is possible to compensate a wrong FoV by moving forward or backward. This holds only approximately if the scene is not perfectly planar. In addition, the effect has hard limits because moving too far will cause objects to move in or out of view, changing the visible skyline. Between these limits, changing the FoV causes both the alignment error and the position to change smoothly. Outside of this stable range, the error is higher, fluctuates more and the position jumps around wildly.

This has two consequences: First, if the FoV obtained from the image's metadata is inaccurate it is usually not a disaster, the retrieved position will simply be slightly inaccurate as well, but not completely wrong. Second, if the FoV is completely unknown, one can get a rough estimate by choosing the minimum error and/or looking for a range where the retrieved position is most stable.

The field-of-view (FoV) extracted from the EXIF data may not always be 100% accurate. This experiment studies the effects of a slight inaccuracy. We modify the FoV obtained from the EXIF by $\pm 5\%$ and plot it against the recognition rate obtained over the entire query set CH1. We observe in Fig. 11(a) that even if the values are off by $\pm 5\%$, we still obtain a recognition rate of $70-80\%$.

Tilt Angle.

Our algorithm assumes that landscape images usually are not subject to extreme tilt angles. In the final experiment evaluated in Fig. 11(b), we virtually rotate the extracted skyline of the query images by various angles in order to simulate camera tilt and observe how recognition performance is affected. As shown in Fig. 11(b) with 30° tilt we still obtain a recognition rate of 60% on the CH1 dataset. This is a large tilt angle, considering that the skyline is usually straight in front of the camera and not above or below it.

Roll Angle.

Our algorithm makes a zero roll assumption, meaning that the camera is held upright. To evaluate the robustness of the algorithm we virtually perturb the roll angle by rotating the extracted skyline of the query image by various angles. Fig. 11(c) shows the achieved recognition rate. For 5° roll angle the recognition rate drops by 26%. This drop does not come as a surprise since the binning of the skyline makes a strong assumption on a upright image. In general this assumption can be relaxed by extending the database with differently rotated skylines, or by using IMU data (often present in today's mobile phones) to correct for the roll angle in the query image. In general we found that landscape images captured with a hand held camera are subject to very little roll rotation, which is also confirmed by both datasets.

Runtime.

We implemented the algorithm partly in C/C++ and partly in Matlab. The segmentation runs at interactive frame rate and gives direct visual feedback to the operator, given the unary potential of our segmentation framework. Given the skyline it takes 10 seconds to find the camera's position and rotation in an area of $40\,000\text{km}^2$ per image. Exhaustively computing an optimal alignment between the query image and each of the 3.5M panoramas would take on the order of several days. For comparison, the authors of [3] use a GPU implementation and report 2 minutes computation time to determine the rotation only, assuming the camera position is already known.

4 Conclusion and Future Work

We have presented a system for large scale location recognition based on digital elevation models. This is very valuable for geo-localization of pictures when no GPS information is available (for virtually all video or DSLR cameras, archive pictures, in intelligence and military scenarios). We extract the sky and represent the visible skyline by a set of contour words, where each contour word is represented together with its offset angle from the optical axis. This way, we can do a bag-of-words like approach with integrated geometric verification, i.e. we are looking for the panorama (portion) that has a similar frequency of contour words with a consistent direction. We show that our representation is very discriminative and the full system allows for excellent recognition rates on the two challenging dataset. On the CH1 dataset we achieve a recognition rate of 88% and 76% on the CH2 dataset. Both datasets include different seasons, landscapes and altitudes. We believe that this is a step towards the ultimate goal of being able to geo-localize images taken anywhere on the planet, but for this also other additional cues of natural environments have to be combined with the given approach. This will be the subject of future research.

Acknowledgements This work has been supported through SNF grant 127224 by the Swiss National Science Foundation. We also thank Simon Wenner for his help to render the DEMs and Hiroto Nagayoshi for providing the CH2 dataset. We also thank the anonymous reviewers for useful discussions and constructive feedback.

References

1. G. Baatz, K. Köser, D. Chen, R. Grzeszczuk, and M. Pollefeys. Leveraging 3d city models for rotation invariant place-of-interest recognition. *International Journal of Computer Vision (IJCV), Special Issue on Mobile Vision*, 96, 2012.
2. G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large scale visual geo-localization of images in mountainous terrain. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 517–530, 2012.
3. L. Baboud, M. Cadík, E. Eisemann, and H.-P. Seidel. Automatic photo-to-terrain alignment for the annotation of mountain pictures. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 41–48, 2011.
4. M. Bansal and K. Daniilidis. Geometric urban geo-localization. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 3978–3985, 2014.
5. J.-C. Bazin, I. Kweon, C. Demonceaux, and P. Vasseur. Dynamic programming and skyline extraction in catadioptric infrared images. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 409–416, 2009.
6. A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 428–441, 2004.
7. M. Brown and D. G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision (IJCV)*, 74:59–73, August 2007.
8. D. Chen, G. Baatz, Köser, S. Tsai, R. Vedantham, T. Pylvanainen, K. Roimela, X. Chen, J. Bach, M. Pollefeys, B. Girod, and R. Grzeszczuk. City-scale landmark identification on mobile devices. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2011.
9. D. Comaniciu, P. Meer, and S. Member. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 24:603–619, 2002.
10. F. Cozman. *Decision Making Based on Convex Sets of Probability Distributions: Quasi-Bayesian Networks and Outdoor Visual Position Estimation*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, December 1997.
11. F. Cozman and E. Krotkov. Position estimation from outdoor visual landmarks for teleoperation of lunar rovers. In *WACV '96*, pages 156 –161, 1996.
12. J. Friedman, T. Hastie, and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. *The Annals of Statistics*, 2000.
13. J. Hays and A. A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, 2008.
14. S. u. Hussain and B. Triggs. Visual recognition using local quantized patterns. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2012.
15. V. Kolmogorov and Y. Boykov. What metrics can be approximated by geo-cuts, or global optimization of length/area and flux. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 564–571, Washington, DC, USA, 2005.
16. L. Ladicky, C. Russell, P. Kohli, and P. Torr. Associative hierarchical random fields. *Pattern Analysis and Machine Intelligence (PAMI)*, 36(6):1056–1077, June 2014.
17. L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *Proceedings of European Conference on Computer Vision (ECCV)*, 2014.
18. J.-F. Lalonde, S. G. Narasimhan, and A. A. Efros. What do the sun and the sky tell us about the camera? *International Journal on Computer Vision*, 88(1):24–51, May 2010.
19. Y. Li, N. Snavely, and D. P. Huttenlocher. Location recognition using prioritized feature matching. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 791–804, 2010.
20. W.-N. Lie, T. C.-I. Lin, T.-C. Lin, and K.-S. Hung. A robust dynamic programming algorithm to extract skyline in images for navigation. *Pattern Recognition Letters*, 26(2):221 – 230, 2005.
21. D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110, 2004.
22. J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and texture analysis for image segmentation. *International Journal of Computer Vision*, 43(1):7–27, June 2001.
23. S. Manay, D. Cremers, B.-W. Hong, A. Yezzi, and S. Soatto. Integral invariants for shape matching. *Pattern Analysis and Machine Intelligence (PAMI)*, 2006.
24. P. C. Naval, M. Mukunoki, M. Minoh, and K. Ikeda. Estimating camera position and orientation from geographical map and mountain image. In *38th Pattern Sensing Group Research Meeting, Soc. of Instrument and Control Engineers*, pages 9–16, 1997.
25. D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 2161–2168, 2006.

26. S. Ramalingam, S. Bouaziz, and P. Sturm. Pose estimation using both points and lines for geo-localization. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, pages 4716–4723, 2011.
27. S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand. Skyline2gps: Localization in urban canyons using omni-skylines. In *IROS 2010*, pages 3816 –3823, oct. 2010.
28. G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *Proceedings of Computer Vision and Pattern Recognition (CVPR)*, pages 1 –7, june 2007.
29. E. Shechtman and M. Irani. Matching local self-similarities across images and videos. In *Proceedings of Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007.
30. J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 1–15, 2006.
31. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *Proceedings of International Conference on Computer Vision (ICCV)*, pages 1470–1477, oct 2003.
32. F. Stein and G. Medioni. Map-based localization using the panoramic horizon. *Transaction on Robotics and Automation*, 11(6):892 –896, dec 1995.
33. R. Talluri and J. Aggarwal. Position estimation for an autonomous mobile robot in an outdoor environment. *Transaction on Robotics and Automation*, 8(5):573 –584, oct 1992.
34. A. Taneja, L. Ballan, and M. Pollefeys. Registration of spherical panoramic images with cadastral 3d models. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 479–486, 2012.
35. W. B. Thompson, T. C. Henderson, T. L. Colvin, L. B. Dick, and C. M. Valiquette. Vision-based localization. In *Image Understanding Workshop*, pages 491–498, 1993.
36. A. Vasilevskiy and K. Siddiqi. Flux maximizing geometric flows. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, pages 1565–1578, 2002.
37. J. Woo, K. Son, T. Li, G. S. Kim, and I.-S. Kweon. Vision-based uav navigation in mountain area. In *MVA*, pages 236–239, 2007.
38. M. Yang, K. Kpalma, and J. Ronsin. A Survey of Shape Feature Extraction Techniques. In P.-Y. Yin, editor, *Pattern Recognition*, pages 43–90. IN-TECH, Nov 2008.

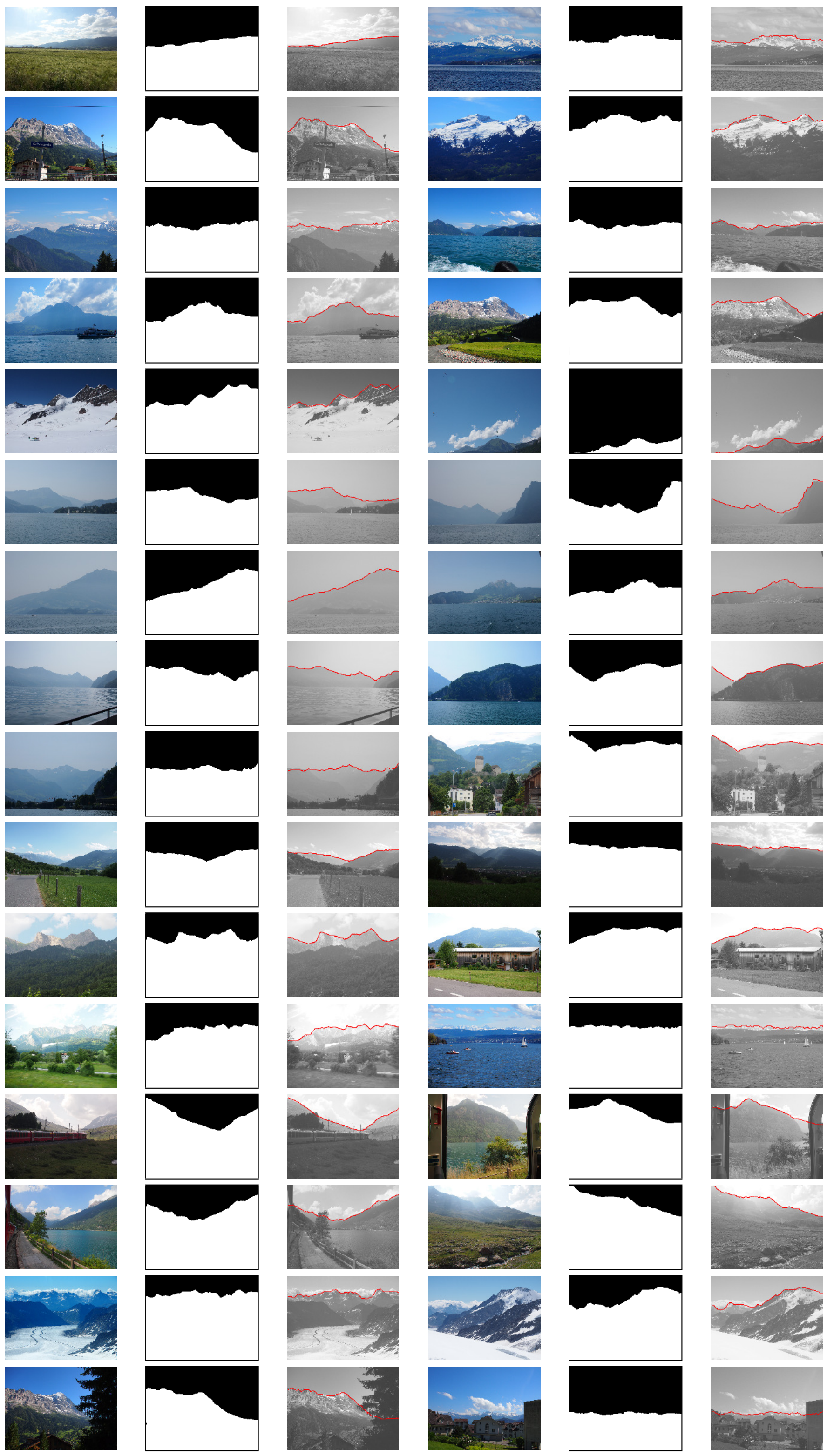


Fig. 8 Sample Results: First and fourth column are input images. Second and fifth column show the segmentations and third and sixth column show the query images augmented with the skyline, retrieved from the database. The images in the last five rows were segmented with help of user interaction.

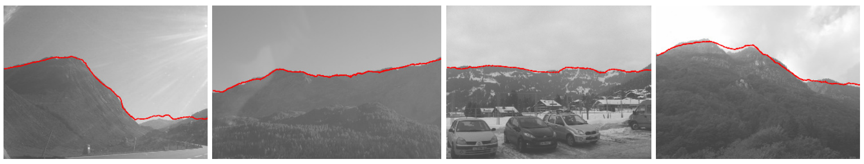
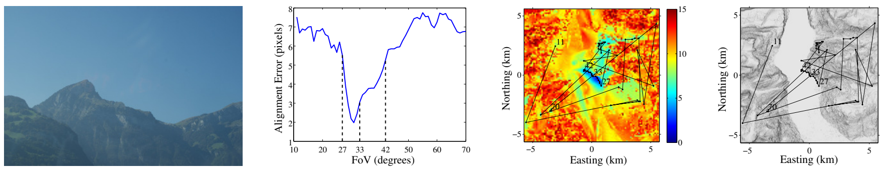
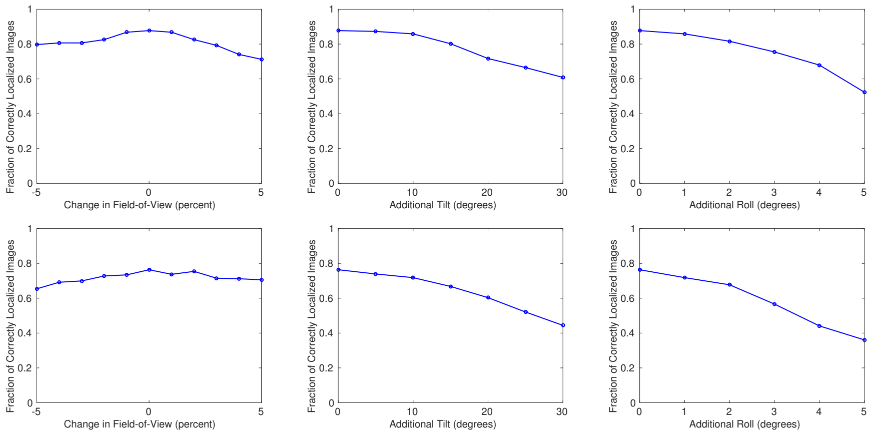


Fig. 9 Some incorrectly localized images. This usually happens to images with a relatively smooth skyline and only few distinctive features. The pipeline finds a contour that fits somewhat well, even if the location is completely off.



(a) (b) (c) (d)

Fig. 10 (a) Query image. (b) Alignment error of the best position for a given FoV. Dashed lines indicate the limits of the stable region and the FoV from the image's EXIF tag. (c) Alignment error of the best FoV for a given position. For an animated version, see `http://cvg.ethz.ch/research/mountain-localization`. (d) Shaded terrain model. The overlaid curve in (c) and (d) starts from the best location assuming 11° FoV and continues to the best location assuming 12°, 13°, etc. Numbers next to the markers indicate corresponding FoV.



(a) (b) (c)

Fig. 11 Robustness evaluation under: (a) varying FoV, (b) varying tilt angle, (c) varying roll angle. Top row CH1 and bottom row CH2 dataset.