2012 Second Joint 3DIM/3DPVT Conference: 3D Imaging, Modeling, Processing, Visualization & Transmission

Leveraging Topographic Maps for Image to Terrain Alignment

Georges Baatz, Olivier Saurer, Kevin Köser, and Marc Pollefeys
Department of Computer Science, ETH Zurich, Switzerland
{*gbaatz|saurero|kevin.koeser|marc.pollefeys*}@*inf.ethz.ch*

Abstract—**With the wide-spread availability of photographic and cartographic data, it becomes desirable to be able to geo-localize any picture in the world. Existing approaches have so far shown impressive results, but they are still lacking in either precision or applicability. In the present work, we explore as an additional cue, semantic image labeling coupled with topographic maps. As an intermediate step towards the ultimate goal of universal geo-localiztion, we show that these cues are suitable for estimating the viewing direction of a terrestrial image, given the image's location.**

I. Introduction

Due to massively available data for many parts of the planet, automatic geo-localization of pictures comes into reach and has recently inspired a lot of works in computer vision [1]–[5]. Among these, the city-based approaches can yield an exact position and orientation because they rely on stable structures (e.g. buildings) that do not change much over time. In contrast, on the countryside the problem is much more difficult and approaches like [2] aim at figuring out a rough neighborhood but cannot determine exact camera parameters in terms of meters or optical axis. An additional complication with countryside footage is, that photo collections (often used as reference material for localization [4]) tend to focus on the major tourist sites in cities but contain little or no material in some less popular areas or hills or forests. Also survey companies that collect street-level (i.e. terrestrial) image data focus on urban areas, meaning that most parts of the countryside are typically only covered from an aerial perspective. In any case, correspondences between an unknown terrestrial image and some older database picture are extremely hard to find, since vegetation changes not only from one year to another but also between seasons, and lighting and weather conditions make it virtually impossible to find similar local image regions in forests, on grass land or dirty roads. However, for virtually all parts of the world topographic maps exist that represent lakes, rivers, roads and railways, forests, settlements and so forth. So, if a major road crosses a river this should give rich information about the viewpoint and orientation of the image. Following this idea, we are interested in the problem of finding the camera pose based on semantic labels in an image rather than trying to exactly match individual pixels. However, it is still extremely challenging to accurately identify the semantics in an image and label each pixel as belonging to one out of many different possible classes reliably. Very often the segmentation varies strongly depending on the training set, training parameters or changes in camera pose, in particular when many classes are involved. Consequently, in this paper, we limit the number of image labels to four. While this increases stability of the labeling, it reduces the discriminative power of the segmentation such that we first focus on the problem of finding the camera orientation given its approximate position. This is an interesting problem in itself, since in recent years, online photo collections[1] have gained a lot of popularity and nowadays millions of photos are available online. Often these photos are geotagged, placed somewhere on a map or have a GPS tag recorded anyways. However, the viewing direction of the photo is typically not available or has to be specified manually and we see this as an excellent problem to demonstrate the feasibility of image to topographic map alignment. In a practical system, this could be one additional source of information that is complementary to e.g. the sun, skylines or other cues.

II. Previous Work

We address the problem of aligning a terrestrial picture with a topographic map given a GPS tag or the map location where a user has dropped the photo. Several authors have targeted the large scale recognition problem based on assumptions about the sky [3] and sun [3], [6], the visible horizon of mountains [7], [8], or satellite weather maps [1] or low level image statistics from photo collections [2], but the results of these approaches are limited to uncertainties in the order of kilometers. Urban approaches can typically localize much more precisely but they assume presence of certain structures like nearby buildings with known 3D geometry [9] or facades that have a sufficient number of stable local image features on them [5], [10].

Maybe the closest work to ours in the literature is the estimation of viewing direction based on a digital elevation model [11]. Here the authors assume to know the camera position and then align edges in the image with ridge and contour features extracted from a 3D model. It is clear that this works best when many characteristic mountains are around and when little edges come from the texture of the scene rather than the geometry. In contrast, in our solution, one relies on the semantic type of terrain that is

[1] e.g. http://flickr.com http://panoramio.com

978-0-7695-4873-9/12 $26.00 © 2012 IEEE
DOI 10.1109/3DIMPVT.2012.33

available in the surrounding. This can easily be adapted to different scenarios (different terrains can be expected in humid or dry regions, in cold or warm areas or at the coast or in the mountains). The problem is somewhat related to those in remote sensing [12]. There, however, the position and orientation is usually well known while the topographic map (or semantic interpretation) of the surfaces needs to be created or updated. In robotics, people have considered indoor semantic localization, however this is mainly treated as a discrete classification problem ("kitchen", "hallway", "office", ...) [13] or based on the (co-)occurence of some classified objects [14], [15]. To the best of our knowledge there has not been any similar attempt on orienting a terrestrial image based on outdoor topographic maps before.

III. Label-based Alignment Method

To align the query image at hand with a topographic map, we have to extract the semantic information, i.e. classify each image pixel as being from one of several classes. Second we warp the topographic map into the predicted position of the terrestrial view for feature based alignment.

Query image: For labeling the query image, we stick to a standard inference model that is based on offline learnt likelihoods for the different semantic classes as well as a smoothness constraint for neighboring pixels. We use the ALE framework of [16] and distinguish the following labels: residential area, bodies of water, sky, and "everything else". This choice is motivated by what types of surface can reasonably be expected to be segmented and what classes of surface are available on the topographic maps. Also, in this contribution we do not focus on improving the state of the art in segmentation and rather use the labeling as a building block on top of which we perform the alignment. The data set used in this paper consists of several locations (outlook point, from a cottage, near a lake, from a hill), where for each location several different images have been taken. The data set is then partitioned into training and test set and the training images are manually labeled. For an example of the segmentation result, see Figs. 7 and 8.

Topographic map: The reference data to align images against is generated from a topographic map and a digital elevation model. We use data obtained from the Swiss Federal Office of Topography, but similar maps are also available for other countries. The topographic data is represented as a vector map. We rasterize it as pixel images with a resolution of one pixel per meter. The chosen labels are mapped to very different colors in order to make them more easily distinguishable at later stages. These images and the elevation model are converted using VirtualPlanetBuilder[2] into a textured 3D terrain model. At the position of interest, which may e.g. be obtained from the query image's EXIF tag, we render the model in each of the main directions. Lighting is deactivated so as to avoid distorting the colors. These perspective views are then combined into one spherical panorama, i.e. x- and y-pixel-coordinates map linearly to yaw and pitch respectively. See Fig. 1 for an example.

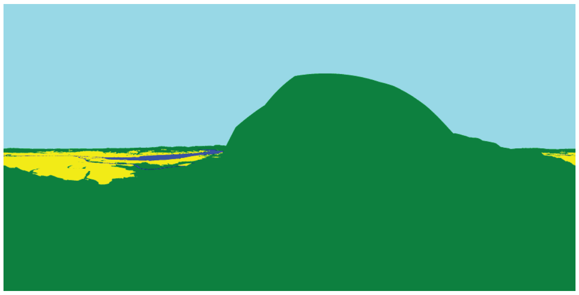


Figure 1. Topographic panorama. Light blue: sky; dark blue: water; yellow: settlements; green: other (mostly vegetation).

A. Alignment

If the camera pose of the query image was an arbitrary 3D rotation, one would have to map both the query image and the panorama to the surface of a sphere and perform the alignment in this space. In practice, however, the rotation is far from arbitrary: As observed in [17], photographers usually try to hold the camera horizontally and they seem to be very good at it. In our experience, roll very rarely exceeds 5 degrees.

Furthermore, pitch usually does not cover the whole range that it theoretically could span: Angles close to ±90° would show only sky or the photographer's feet. To get a feeling for how much a seemingly small pitch of e.g. 20° is, consider a normal camera lens (meaning neither wide-angle nor tele). As a rule-of-thumb in photography, the focal length is approximately equal to the image diagonal [18] which corresponds to a diagonal field-of-view of about 53°. For a 4:3 landscape image format this induces a 33° vertical field-of-view. This means that already for a pitch of ±16.5°, objects that are straight in front of the user are just barely visible at the top or bottom edge of the image. With a pitch larger than that, these objects would move completely out of view. Therefore, with the exception of special cases (like e.g. the photographer standing very close to and looking up along a tall object), the pitch tends to be at most a few tens of degrees.

The fact that pitch tends to be reasonably small, and roll very small can be used to simplify the alignment problem considerably. One such simplification is the use of spherical panoramas for alignment. In theory, yaw corresponds to a horizontal translation in the panorama, while pitch and roll correspond to more complicated non-rigid transformations. Fortunately those angles are expected to be small, so roll and pitch can be approximated quite well as rotation and vertical translation respectively. See Fig. 2 for a comparison.

[2] http://www.openscenegraph.org

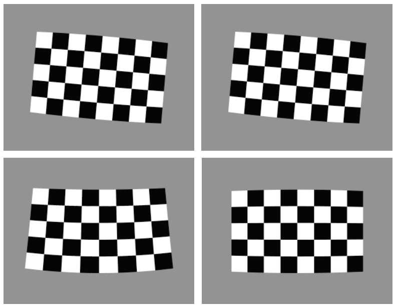
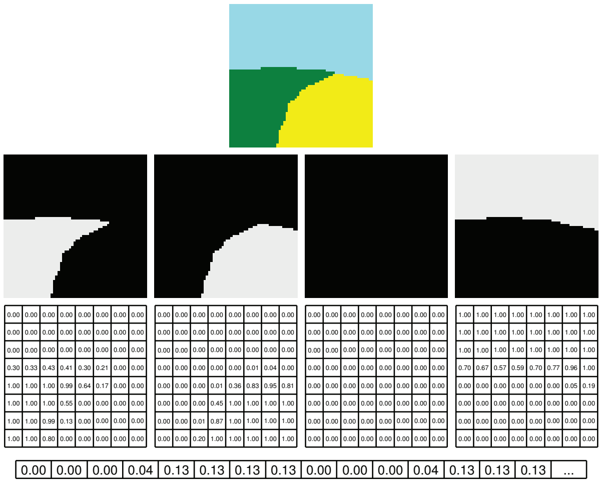


Figure 2. Comparison between the true mapping and our approximation. Left column: exact warping. Right column: approximate rigid transformation. Top row: 5° roll. Bottom row: 20° pitch.



0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.30	0.33	0.43	0.41	0.30	0.21	0.00	0.00
1.00	1.00	1.00	0.99	0.64	0.17	0.00	0.00
1.00	1.00	1.00	0.55	0.00	0.00	0.00	0.00
1.00	1.00	0.99	0.13	0.00	0.00	0.00	0.00
1.00	1.00	0.80	0.00	0.00	0.00	0.00	0.00

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.01	0.04	0.00
0.00	0.00	0.00	0.01	0.36	0.83	0.95	0.81
0.00	0.00	0.00	0.45	1.00	1.00	1.00	1.00
0.00	0.00	0.01	0.87	1.00	1.00	1.00	1.00
0.00	0.00	0.20	1.00	1.00	1.00	1.00	1.00

0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
0.70	0.67	0.57	0.59	0.70	0.77	0.96	1.00
0.00	0.00	0.00	0.00	0.00	0.00	0.05	0.19
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

0.00	0.00	0.00	0.04	0.13	0.13	0.13	0.13	0.00	0.00	0.00	0.04	0.13	0.13	0.13	...

Figure 3. Calculation of a feature descriptor. Top: Image patch from which the desciptor is calculated. Row 2: Binary images corresponding to one label each. Row 3: Weighted averages over their respective tiles. Bottom: Final (normalized) feature desciptor.

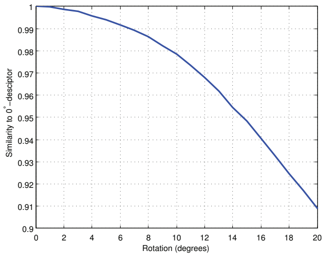


Figure 4. Robustness of the proposed feature desciptor with respect to rotation. The image patch from Fig. 3 was used to generate this curve. Similarity (as defined in Section III-C) to the 0°-desciptor remains above 0.99 for angles up to 6°. Thus, the maximum roll of 5° we expect is well accomodated.

B. Feature descriptor

Rather than naively finding the transformation which maximizes the number of matching pixels between the label map of the query image and the panorama, we choose to use feature descriptors that we designed for this purpose. This increases efficiency and, more importantly, robustness against large non-matching areas. Such mismatches may be due to misclassifications in the segmetation stage or objects (such as trees or buildings) that are missing from the model. Objects that are close to the camera cover a large portion of the image and may introduce a big mismatching area. See for example Fig. 7(b), where the trees in the foreground induce a large green area that does not exist in the panorama.

If we were just optimizing for matching pixels, such an area would unduly pull the optimum away from the true alignment. We mitigate this effect by encoding only the *boundaries* between uniform areas. This way, in case of a mismatching area, we penalize only for the boundary rather than the entire area.

In order to compute a feature, each of the ℓ labels is treated separately. The calculations are based on a binary image indicating for each pixel whether or not it has been given the label under consideration. The desciptor depends on a square image region which is further subdivied into $k \times k$ square tiles of size $w \times w$ pixels each. For each such tile, we evaluate the average weighted by a gaussion distribution with the tile's center as mean and standard deviation $\sigma = {}^{w}\!/\!{}_{2}$. This results in ℓk^2 numbers total (over all labels), which are gathered into one vector. The final descriptor is obtained by scaling the vector such that it has unit norm. See Fig. 3 for an illustration.

We deliberately designed the descriptor to be not rotationally invariant in order to leverage the fact that roll tends to be small. This makes the desciptor more discriminative. Nevertheless, we ensured that it is robust to small rotations, see Fig. 4.

Using a keypoint detector has in our case two main disadvantages: 1) Stable points (i.e. corners and junctions) are not very frequent and 2) since the precise shape of the boundaries often differs between the model and the query image (see Fig. 7), it is unlikely that the same set of keypoints will be detected in both images. We resolve this issue by computing features densely, at steps of d degrees. The glut of features is reduced by discarding those that correspond to almost uniform areas. More precisely, we define the non-unifomity of a feature vector f (before normalizing) as

$$v(f) = \max_{i=1}^{\ell} \left(\max_{j=(i-1)k^2+1}^{ik^2} f_j - \min_{j=(i-1)k^2+1}^{ik^2} f_j \right) \tag{1}$$

and reject all features with $v(f)$ less than some threshold $t_{\text{var\_min}}$. The intuition is that for at least one of the labels, the numbers should vary by at least $t_{\text{var\_min}}$ for a descriptor to be admitted. This strategy not only retains features at corners and junctions, but also those describing edges, which (although they don't provide one-to-one correspondences) still provide useful information for the alignment. See Fig. 5.

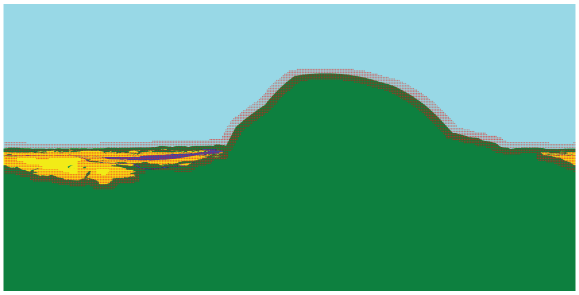


Figure 5. Red dots mark the locations of feature points after rejecting nearly uniform features. Note how only features near boundaries are retained.

C. Hough transform

We determine the optimal alignment using a hough transform. Prior knowledge about small roll and pitch can be used to significantly reduce the parameter space. We consider all the pairs of features with a similarity above some threshold $t_{\text{sim\_min}}$ to be a putative feature correspondence. As similarity measure, we use the euclidian scalar product between the feature vectors.

$$s(f, f') = \langle f, f' \rangle \tag{2}$$

This way, the complete similarity matrix can be very efficiently calculated as a matrix product.

Every putative feature correspondence adds a vote to all of the compatible yaw/pitch/roll combinations. Let the feature points be given as (x, y) in the panorama and (x', y') in the query, then α (yaw), β (pitch) and γ (roll) must fulfill these equations:

$$x = \alpha + x' \cos\gamma - y' \sin\gamma \tag{3}$$

$$y = \beta + x' \sin\gamma + y' \cos\gamma \tag{4}$$

For any fixed x, y, x', y', the solution is a 1-parametric family. The equations can easily be solved for α and β, and then one can iterate over the values of γ to generate solutions.

The weight of a putative feature correspondence (f, f') depends on two things: uniqueness and similarity. Let F be the set of all features in the panorama. We define the entropy coefficient

$$c_{\text{ent}} = \log \frac{|F|}{|\{f'' \in F : s(f'', f') \geq t_{\text{sim\_min}}\}|} \tag{5}$$

This coefficient gets bigger, the fewer “similar” features f' has, i.e. the more *unambiguously* it matches.

Secondly, we define the similarity coefficient

$$c_{\text{sim}} = \min\left(\frac{s(f, f') - t_{\text{sim\_min}}}{t_{\text{sim\_max}} - t_{\text{sim\_min}}}, 1\right) \tag{6}$$

for some upper similarity threshold $t_{\text{sim\_max}}$. This value is clamped from above to 1 and it cannot become negative since we only consider feature pairs with similarity at least $t_{\text{sim\_min}}$. This coefficient gets bigger, the *stronger* f' matches.

A putative feature correspondence adds a vote with weight equal to $c_{\text{ent}} \cdot c_{\text{sim}}$ to each of the compatible yaw/pitch/roll combinations. This way, our voting scheme favours unambiguous and strong matches, while still making use of weaker and more ambiguous ones. See Fig. 6 for an example of the Hough transform.

IV. Experiments

A. Parameter selection

For the feature desciptor, $\ell = 4$ is required by our choice of labels. We further set $k = w = 8$, leading to 256-dimensional desciptors which is a common size. We render (full and partial) panoramas at 8 pixels/degree, that way one tile of a descriptor covers exactly $1°$. Features are computed at steps of $d = 1°$. We set $t_{\text{var\_min}} = 0.5$, the exact value of this parameter has little influence, since the gaussian PDF decreases very quickly.

The parameter space for the Hough transform is discretized at a resolution of $1°$ in order to match d. For pitch, we keep the full range of $-90° \ldots 90°$ since this does not negatively impact efficiency; roll, however, does and is therefore restricted to $-5° \ldots 5°$. Finally, we set $t_{\text{sim\_max}} = 1.0$ (the maximal value that makes sense) and $t_{\text{sim\_min}} = 0.5$; we can afford going so low because the similarity coefficient ensures that weak matches do not get undue weight.

B. Results

The queries are chosen to be images showing all 4 labels. This increases the chances that they contain enough information for orienting the image.

Fig. 6 shows the Hough transform for Fig. 7(a). The vertical alignment (pitch) is very strongly constrained, the horizontal one (yaw) not so much. This is due to the fact the the images contain mostly horizontal lines. Zero roll is clealy preferred over more extreme values.

Fig. 7 shows some successful results. Note the difficulty of the problem: Far-away mountains tend to blend with the sky which makes segmentation very challenging. The topographic maps contain very high geometric details (like thin rivers or jagged boundaries), which are not recognizable in the query images. Thus one can not rely on small details for alignment. Some images, like (a-b) and to a lesser extent (c), show foreground trees and hedges that are not present in the panorama, so sometimes not even the coarse structures match. An alignment algorithm needs to be robust against all these effects.

Fig. 8 shows some unsuccessful results. For image (a) the rough direction is about correct, but its sighlty too far right. This may be due to the slight over-extension of the lake's left in end (in the segmentation) which got matched to the right end (in the panorama). For image (b), only the closest

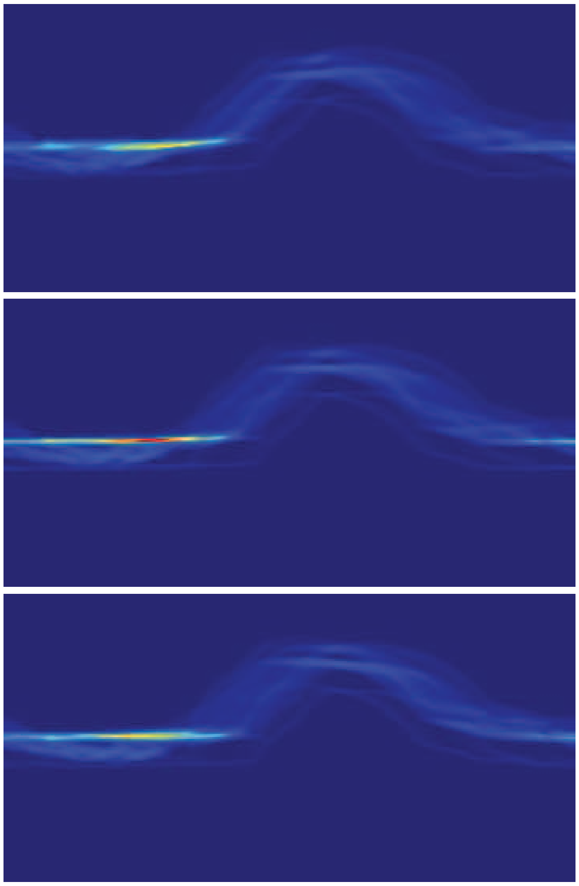


Figure 6. Some slices through the Hough space for Fig. 7(a) corresponding to different values of roll. From top to bottom: $-5°$, $0°$, $5°$. Blue: zero score, red: maximum score.

mountain range is visible and the rest is vanishing in fog. Therefore, the detected horizon is lower than it should be. Both images have a smaller field-of-view and thus fewer features compared to those of Fig. 7, which also makes alignment more difficult.

V. Conclusion and Future Work

We have presented a method for aligning images to terrain based on topographic maps. While there is still some room for improvement in both the segmentation and the alignment stage, we have demonstrated promising results especially on images that are unlikely to be correctly aligned by earlier approaches based on the horizon or visible mountain ridges, due to lack of these features. It would be intersting to make a combined system that uses all of these cues together so that it is applicable to a wider range of images. Furthermore, it seems natural to extend the present method so that in addition to orientation, it also estimates location and possibly even the camera's intrinsics.

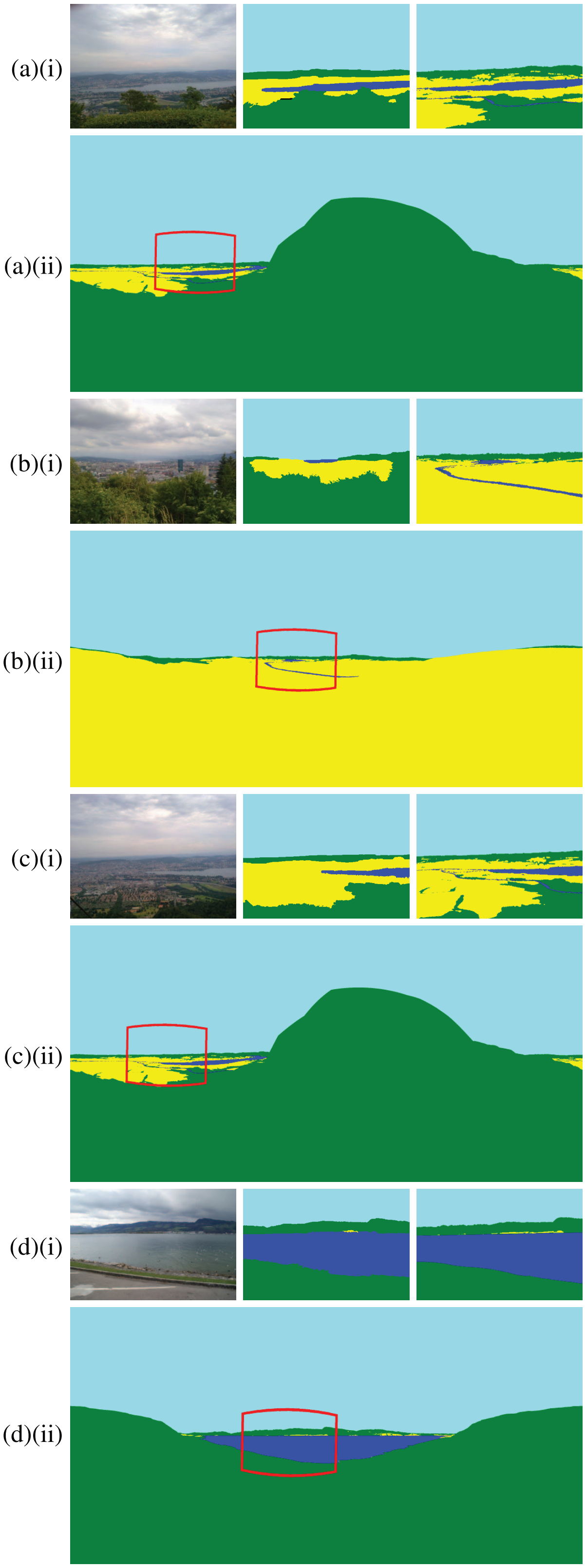


Figure 7. Some successful results. Rows (i): Left: Query image. Middle: Segmentation. Right: View of the terrain model rendered from the estimated viewpoint. Rows (ii): Panorama with estimated viewpoint overlaid in red.

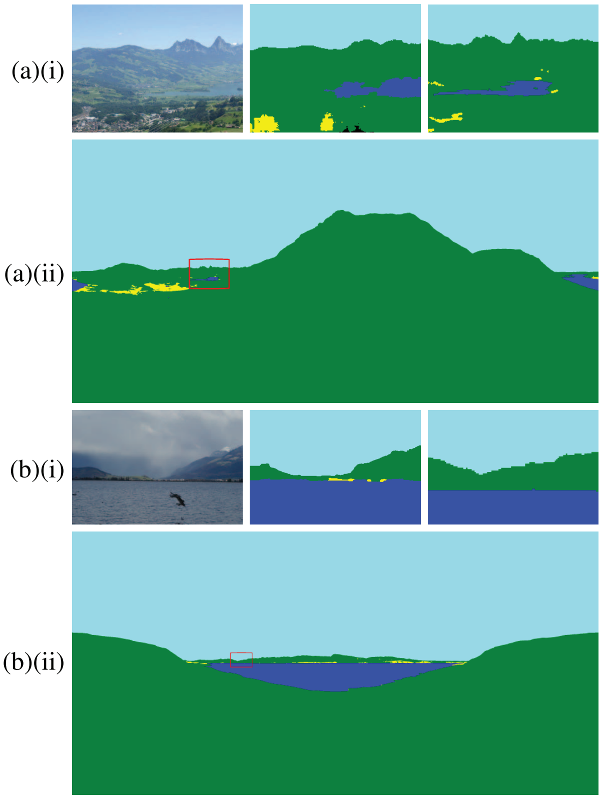


Figure 8. Some unsuccessful results. Rows (i): Left: Query image. Middle: Segmentation. Right: View of the terrain model rendered from the estimated viewpoint. Rows (ii): Panorama with estimated viewpoint overlaid in red.

Acknowledgment

This work has been supported through SNF grant 127224 by the Swiss National Science Foundation.

References

[1] N. Jacobs, S. Satkin, N. Roman, R. Speyer, and R. Pless, "Geolocating static cameras," in *IEEE International Conference on Computer Vision (ICCV)*, Oct. 2007.

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

[3] 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*, vol. 88, no. 1, pp. 24–51, May 2010.

[4] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *Proceedings of the 11th European conference on Computer vision: Part II*, ser. ECCV'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 791–804. [Online]. Available: http://dl.acm.org/citation.cfm?id=1888028.1888088

[5] 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 *CVPR 2011*, 2011.

[6] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan, "Estimating the natural illumination conditions from a single outdoor image," *International Journal of Computer Vision*, 2011.

[7] F. Cozman and E. Krotkov, "Position estimation from outdoor visual landmarks for teleoperation of lunar rovers," in *Applications of Computer Vision, 1996. WACV '96., Proceedings 3rd IEEE Workshop on*, 1996, pp. 156 –161.

[8] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys, "Large scale visual geo-localization of images in mountainous terrain," in *Lecture Notes in Computer Science (Proceedings of ECCV 2012)*, 2012.

[9] S. Ramalingam, S. Bouaziz, P. Sturm, and M. Brand, "Skyline2gps: Localization in urban canyons using omni-skylines," in *Intelligent Robots and Systems (IROS), 2010*, oct. 2010, pp. 3816 –3823.

[10] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR '07*, june 2007, pp. 1 –7.

[11] L. Baboud, M. Cadík, E. Eisemann, and H.-P. Seidel, "Automatic photo-to-terrain alignment for the annotation of mountain pictures," in *CVPR*, 2011, pp. 41–48.

[12] T. M. Lillesand, R. W. Kiefer, and J. W. Chipman, *Remote Sensing and Image Interpretation*. Wiley, 2004. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/0471152277

[13] C. Yi, I. H. Suh, G. H. Lim, and B.-U. Choi, "Active-semantic localization with a single consumer-grade camera," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, oct. 2009, pp. 2161 –2166.

[14] A. Nüchter and J. Hertzberg, "Towards semantic maps for mobile robots," *Robot. Auton. Syst.*, vol. 56, no. 11, pp. 915–926, Nov. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.robot.2008.08.001

[15] H. Lim and S. Sinha, "Towards real-time semantic localization," in *ICRA Workshop on Semantic Perception*, 2012.

[16] L. Ladicky, C. Russell, P. Kohli, and P. H. Torr, "Graph cut based inference with co-occurrence statistics," in *European Conference on Computer Vision*, 2010.

[17] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *Int. J. Comput. Vision*, vol. 74, pp. 59–73, August 2007. [Online]. Available: http://dl.acm.org/citation.cfm?id=1265138.1265141

[18] L. D. Stroebel, *View Camera Technique*. Focal Press, 1999. [Online]. Available: http://books.google.ch/books?id=71zxDuunAvMC