Automated Silhouette Extraction for Mountain Recognition

Daniel Braun
Heinrich-Heine-Universität
Institut für Informatik
Universitätsstraße 1
40225 Düsseldorf, Deutschland
braun@cs.uni-duesseldorf.de

Michael Singhof
Heinrich-Heine-Universität
Institut für Informatik
Universitätsstraße 1
40225 Düsseldorf, Deutschland
singhof@cs.uni-duesseldorf.de

ABSTRACT

With the rise of digital photography and easy sharing of images over the internet, a huge amount of images with no notion of what they are showing exists. In order to overcome this problem, we – for the example of mountain recognition – introduce a method, that is able to automatically recognise a mountain shown in a photography.

Our method does not require GPS information stored in the image, since most images are not GPS tagged, either because of the absence of a GPS sensor in the device or because it has been deactivated for a lesser power consumption, which often is the case with smartphones. Instead, we propose a method that is able to automatically extract the mountain's silhouette from a given image. This silhouette is then cleaned by removing artefacts and outliers, such as trees and clouds, with a histogram based approach. Finally, the cleaned silhouette can be compared to reference data in order to recognise the mountain that is shown in the picture. For this, time series comparison techniques can be used to find matching silhouettes. However, because of the huge number of reference silhouettes to compare against, we argue, that a preselection of those silhouettes is necessary and point out approaches to this problem.

Categories and Subject Descriptors

I.4.8 [**IMAGE PROCESSING AND COMPUTER VISION**]: Scene Analysis—*Object recognition*; I.4.6 [**IMAGE PROCESSING AND COMPUTER VISION**]: Segmentation—*Edge and feature detection*; H.2.8 [**DATABASE MANAGEMENT**]: Database Applications —*Data mining*

Keywords

Object Recognition, Image Annotation, Outlier Detection, Image Segmentation, Skyline Recognition, Mountain Recognition, Time Series

27th GI-Workshop on Foundations of Databases (Grundlagen von Datenbanken), 26.05.2015 - 29.05.2015, Magdeburg, Germany.

1. INTRODUCTION

Sharing our experiences with digital images is a significant part of our today's life, which is partly a result of the high availability of digital cameras, like in smartphones, and the high use of social networks, that simplifies the publication and sharing of images. As a consequence, the number of images in the world wide web increases significantly. For example, this can be seen on the image sharing platform Instagram, where users share an average of 70 million new photos per day [1].

As a result of such high numbers of images, searching photos which show specific objects is challenging, because the majority of these images is not properly tagged with the names of every object seen in them. So the need for efficient algorithms for automatic object recognition rises. In the last decades there were many advances in this research field, but especially the automatic identification of landmarks, which are subject to weather changes, areal erosion and vegetation, is still a challenging task, even if the amount of images with attached GPS data, which marks the geo-position of the camera when the photo was shot, is rising.

The growing spread of devices with the capability of generating GPS tags for the images, like smartphones and digital cameras with GPS units, enables many possibilities for an subsequent geo-localisation of images, due to the fact that GPS tags can significantly reduce the number of possible sights, buildings or landmarks to compare with. However, there exist too many images without the advantage of GPS tags, so that an automatic geo-localisation without prior knowledge of the camera position is still a valuable aim.

Our focus lies on the automatic landmark recognition, which we will describe by using the example of mountain recognition in images. To solve the question, which mountain can be seen on an given image, we match the skyline of the mountain in the image with silhouettes of mountains in our database. For this purpose, we have to automatically extract the exact skyline from the image, which is a difficult task, because the segmentation of the image can lead to artefacts, for instance due to weather conditions, noise, obstacles or overexposure.

In this paper we introduce a baseline segmentation algorithm, which uses an outlier detection algorithm to identify and eliminate these artefacts. The article is structured as follows: In the next section we discuss other papers related to our algorithm. We then introduce our algorithm, which consists of the three steps silhouette extraction, silhouette cleaning and silhouette matching. The section of the latter one will be a perspective of how to use the cleaned silhouette in further steps. The last chapter summarises this paper and outlines future work.

2. RELATED WORK

The amount of publications for automated mountain recognition increased significantly in the last two decades. Given a high number of publicly available digital elevation maps (DEM), the consensus in many publications [2, 3, 4, 5, 10] is to use image-to-model matching for mountain recognition and orientation identification.

Many approaches, like [3, 4, 10], make use of known GPS data for the image to align in the terrain. This reduces the search space for possible mountain silhouettes significantly, because the search of the correct mountains is limited in checking the surrounding of the camera's position.

Baboud et al. [3] need an estimated field-of-view to calculate the rotation which maps the image to the terrain. Therefore, the authors introduce a robust matching metric, using extracted edges in combination with a search space reduction to further reduce computation time. [10] uses a standard Sobel-filter for the edge extraction. To identify the edges which are part of the mountain's contours, the authors propose the use of the Random Ferns classifier. Afterwards they match the emerged contour map with the contour map extracted from the DEM. In [4] a 360-degree panorama of the surroundings of the image location is synthesized out of a given DEM and used for matching the skyline extracted from the image. For that, they use a vector cross correlation (VCC) technique to find the candidate matches. After further refinement and recalculation of the VCC for each peak they can label the peaks with a high precision.

All three approaches show good results for their issue, but the need for GPS data for the processed image does not match our problem specifications, different from Baatz et al. [2], in which the focus lies on images without GPS tag. They use an approach based on a cost function for the belonging of a pixel to the sky respectively foreground. They combine this approach with a relevance feedback-like user interference, where the user can mark parts of the sky or the foreground. This user intervention was needed for 49% of the images in their dataset, which was collected during there research. Thankfully, they published this dataset, so that it will be used in this paper. After the contour extraction, they match the coded contourlet with the contours extracted from a DEM at several points on a predefined grid. At last, when they find a suitable match, they recalculate the geo-position of the camera. Naval et al. [5] also try to find the camera position and orientation, using a DEM to position the camera on the world. For this purpose they match the extracted skyline from an image with a synthetic skyline from a DEM. Different to both works we try to get an automatically cleaned silhouette, thus removing obstacles or other artefacts, out of the processed image.

Tao et al. [12] focus on the identification of the sky seen in an image and the search for images with a specific sky appearance. Therefore they define different sky attributes, like for example the sun position, which they extract individually afterwards. At last they present their complete system SkyFinder, in which an attribute based search is implemented. On top of that, they provide a sky replacement algorithm for changing the sky in an image. However, the recognition of the skyline is not part of this system.

3. SILHOUETTE RECOGNITION

The silhouette recognition process is basically a process in three steps. The first step is the extraction of the silhouette from a given picture. This silhouette is stored as a polygonal chain. During the second step, this silhouette is cleaned by identifying and removing outliers. The cleaned silhouette is then used as the input to the third and final step, which consists of matching the silhouette against the reference data in order to be able to identify the structure in the picture.

3.1 Silhouette Extraction

Extracting the silhouette is the first step in the mountain identification process described in this paper. The task of this step is the separation of the sky from the rest of the processed image. We therefore have a binary segmentation task to solve, in which we check every pixel p of an image and decide if p shows a part of the sky or not. For a human this would be easy to do in most cases, even though it would be too much work and time to segment great numbers of pictures manually. Because of that, we use a growing seed algorithm, like the algorithm described in [11], for which we use the fact, that in most cases the pixels at the upper bound of the image are part of the sky. In the following section, we will first describe some difficulties, which can make the segmentation task more complex. After that, our baseline segmentation algorithm will be further explained.

The segmentation of an image generally suffers from different problems, like under-/overexposure, which results in a smaller difference between the pixel colours, or blur, which can be, for example, a result of a lossy compression of the image. In addition, our binary segmentation task has even some own problems to deal with. The first difficulty is a consequence of the motif itself, because the weather in mountainous terrain is very volatile. This and the fact that, for example in the alps, many mountain peaks are covered with snow, can make the extraction of the mountain silhouette out of an image imprecise. Furthermore differently coloured bands of cloud can lead to an extracted silhouette which lies in the sky and is therefore not part of the real skyline. The second one is a result of possible obstacles, which hide the real silhouette of the mountain or lead to smaller segments within the sky segment.

Even though we are aware of these problems, our naive segmentation algorithm cannot handle all of them. For the identification of artefacts as result of segmentation errors, we use the second step of our chain, which is described in section 3.2. Our segmentation algorithm uses the upper row of pixels as seed for the sky segment. This means that we mark the upper pixels as sky and process every neighbouring pixel to let this segment grow. For this purpose we first convert the processed photo to a gray-scale image G. Now we can define V_G as overall variance of the brightness values. Having V_G, we now process every pixel $p_{(x,y)}$ which is not a part of the sky segment and mark it as sky candidate, if for $p_{(x,y)}$ it holds that

$$B_{p_{(x,y)}} - mean^r_{p_{(x,y)}} < \gamma \cdot \sqrt{V_G}$$

with $B_{p_{(x,y)}}$ the brightness of the pixel $p_{(x,y)}$, $mean^r_{p_{(x,y)}}$ as the mean of the brightness in an neighbourhood of the pixel with the radius of r and γ as a factor to scale the impact of the standard derivation. This means that we mark a pixel, if its distance to a local mean of brightness values

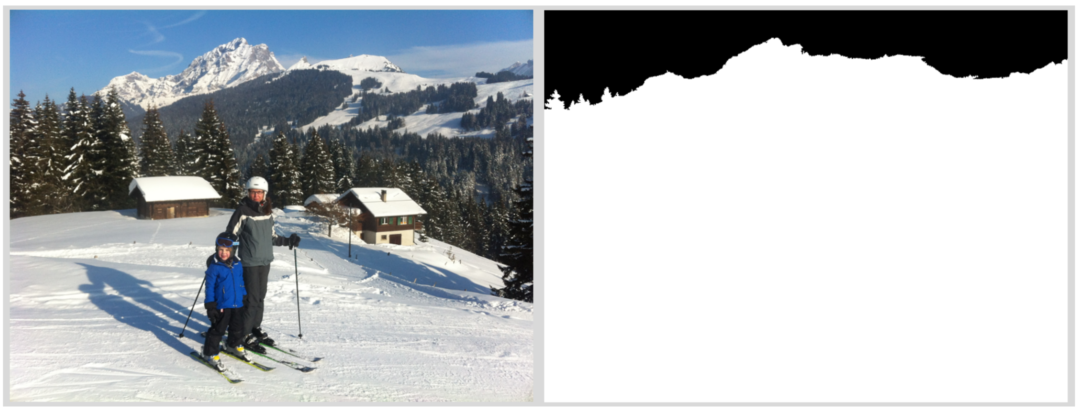

Figure 2: The result of the segmentation algorithm. (Left) The original image out of the dataset from [2]. (Right) The binary image after the segmentation. White pixels mark the ground segment.

is smaller than a fixed percentage of the global standard derivation of all brightness values. The idea behind this term is, that the border between sky and ground has in most cases a stronger contrast than the rest of the image, especially the sky. Therefore, the distance to the mean will be higher at the skyline as it will be at the border of possible clouds. With the connection of the upper bound to the standard derivation, we want to take into account the images where the brightness is very homogenous, for example due to overexposure, and therefore the contrast of the skyline decreases.

In our experience this naive assumption shows good results for $r = 5$ and $\gamma = 0.1$ on most images out of the swiss dataset published in [2]. In the future we will test more complex algorithms, like for instance the skyline extraction algorithm proposed in [8], with the expectation that they will yield even better results. After we have marked all possible candidates, we can finally let the sky segment grow. For that, we check for every pixel $p_{(x,y)}$, if it is a 4-connected neighbour, for explanation see the left side of figure 1, to a pixel of the sky segment and marked as sky candidate (see the previous step). If so, the pixel will get marked as sky. This step is repeated until no more pixels can be added to the sky segment. At this point, we have a binary image, which represents the skyline as transition between the sky and the ground, like the one shown in figure 2.

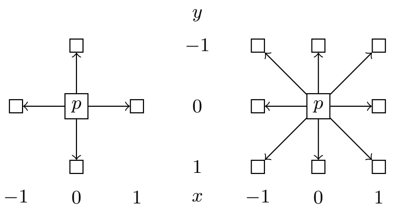

Figure 1: Two possible neighbourhoods of a pixel. (Left) 4-connected neighbourhood of the pixel p. (Right) 8-connected neighbourhood of the pixel p.

For the silhouette extraction, we search a silhouette $S = (v_1, \dots, v_n)$ where every vertex $v_i = (x_i, y_i)$ is a two dimensional point where x_i describes the x-coordinate and y_i describes the y-coordinate of that point in the picture. We start at the upper left pixel $p_{(0,0)}$ and search for the first non-sky pixel as start vertex v_1 for our polygonal chain, which lies on the left pixel column with $x_1 = 0$. Now, we have two possibilities: First, we can find no pixel, which results in checking the next column until we find a pixel or we have reached the lower right pixel of the image. Otherwise, if a possible skyline pixel was found, the algorithm tracks the transition between the sky and the ground in the following manner.

Having v_i as the last extracted vertex of the silhouette, the algorithm checks the 8-connected neighbours of this point (see the right side of figure 1) and chooses the non-sky point as next vertex which has the lowest angle between the incoming line, defined by the two last vertices v_{i-1} and v_i, and the outgoing line, defined by v_i and the neighbouring point. This can easily be done, as shown in figure 3, by checking the 8-connected neighbours from v_i in clockwise direction, starting at the predecessor. The only exception is the start point v_1, where we set $v_0 = (-1, y_1)$. This means that we choose in this case the left direct neighbour, which lies outside of the image, as predecessor.

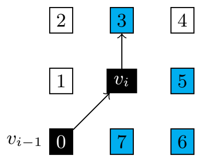

Figure 3: An example how to choose the next silhouette point. Starting at v_{i-1} as predecessor of the current vertex v_i, the algorithm searches in clockwise direction for the next ground pixel (blue). Therefore the third pixel will be chosen.

Now, we have the border of the mountain as chain of neighbouring pixels. To reduce the amount of vertices without loosing any information, we eliminate all vertices which bring no further information gain to the final silhouette in the last step of the extraction phase. For this, we define the

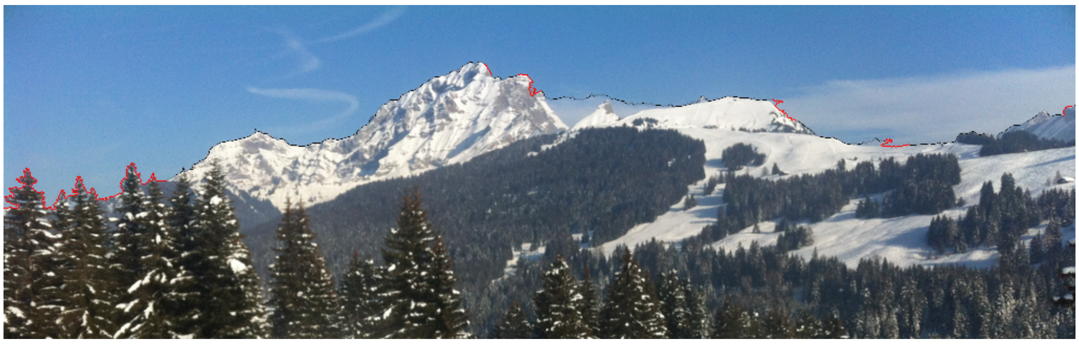

Figure 4: Silhouette as black line with outliers in red on original image.

information gain I of a vertex v_j, with $1 < j < n$, as

$$I_{v_j} = \begin{cases} 1, & \text{if } \angle v_{j-1}v_jv_{j+1} \neq 180^\circ \\ 0 & \text{otherwise.} \end{cases}$$

This means that every vertex which lies on one line with his predecessor and successor carries no further information for the extracted silhouette. After deleting every vertex with $I_{v_j} = 0$, we obtain our final polygonal chain, which can now be analyzed in the the cleaning phase described in the following section.

3.2 Silhouette Cleaning

The extracted silhouette from the previous step may contain different disturbances. In this step, we try to get rid of those, so that they cannot affect the step of silhouette matching that gets described in section 3.3. Examples for such disturbances are manifold, ranging from objects in front of the mountain, such as trees, to errors produced during the extraction process that can be caused by a low contrast between the mountain line and the sky. Essentially, after finding an outlier, we currently cut elevations within this part of the silhouette away.

Some of such outliers are showcased in figure 4. Here, the black line is the silhouette that has been extracted by the extraction step as described in section 3.1. The red parts are those parts of the silhouette, that are automatically detected as outliers. Examples for outliers are the tree tops to the left, that have correctly been marked as outliers. In the center of the picture, there is a part of the mountain, that has been cut out. Here the contrast between rock and blue sky gets to low for the extraction step to distinguish them. Right next to this, clouds around the lower peak in the back get recognised as mountain. Since the outline of this outlier is not atypical for a mountain silhouette, this outlier is not recognised. Farther to the right, there are two other outliers where clouds are recognised as mountain by the extraction step. These are, in this case, marked red and such detected by the silhouette cleaning step.

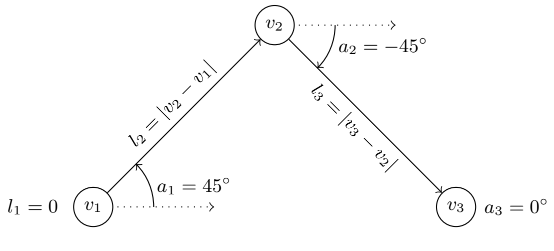

Figure 5: Conversion of polygonal chain for better pattern recognition.

Note, that there are a few parts of the silhouette that get marked as outliers but are not described above. These are false positives. However, since the image here is used to showcase the different kind of outliers, these are not discussed here.

The silhouette cleaning part of our algorithm is, again, divided into three parts. As an input, it gets the silhouette $S = (v_1, \dots, v_n)$ extracted from a picture.

For the recognition of patterns inside the polygonal chain, this representation has disadvantages, because all vertices are given in absolute positions. For outlier detection, a model where similar structures do look similar in their representation is beneficial. Therefore, we convert S to a representation $AP = (ap_1, \dots, ap_n)$, with $ap_i = (l_i, a_i)$. Here, we set

$$l_i = \begin{cases} |v_i - v_{i-1}|, & \text{if } i > 1 \\ 0 & \text{else} \end{cases}$$

and a_i as the angle between the vector $v_{i+1} - v_i$, and the x-axis for $i < n$ and $a_n = 0^\circ$. Figure 5 illustrates this. During this step, points where the angle does not change are removed. Also, artefacts consisting of angles of 180° between two following segments get removed.

The basic idea of the outlier detection method itself is to compare histograms created from parts of the polygonal chain AP to a reference histogram. The latter hereby is created from the silhouettes of the ground truth data as given in [2].

The mentioned histograms consist of the two dimensions segment length and angle size, just as the points in the transformed polygonal chain AP, and are normalised. This means, that the sum of the frequencies of each bucket is 1. For the reference data, we compute one histogram from each image's silhouette. Finally, the mean of all these histograms is computed and becomes the reference histogram H_r.

In order to find outliers, we use a sliding window approach over the polygonal chain AP_i of a given input image. We therefore use a section of m successive segments $ap_i, \dots, ap_{i+m-1}$ in order to compute a histogram H_i with the same bucket distribution as in H_r. Then, for every point used to compute H_i, we store the distance between H_i and H_r. By this approach, for most points, multiple distance values get stored when the sliding window is moved on. The final distance d_i for a point ap_i is now computed as the average distance of all distances stored for this point.

As distance function in this case we use the one given in the following

Definition 1. *Let $G = (g_1, \ldots, g_k)$, $H = (h_1, \ldots, h_k)$ be normalised histograms with the same bucket distribution.*
The above average distance of G to H is defined by

$$D(G, H) := \max(|aab(G)|, |aab(H)|) - |aab(G) \cap aab(H)|,$$

where

$$aab(F) := \left\{ i \in \{1, \ldots, k\} \,\middle|\, f_i \geq \frac{1}{k} \right\}$$

with F being a normalised histogram with k buckets.

This can be implemented to compute in linear time to the number of buckets, or, for histograms of a fixed length, in constant time.

Based on a number of images used as training data, in the same way as described above, the medium distance μ and the standard deviation σ can be determined. We use these precomputed values in order to find two thresholds $\tau_{in} > \tau_{out}$, such that we can characterise a point ap_i as a strong anomaly if it holds that its distance

$$d_i \geq \mu + \tau_{in} \cdot \sigma$$

and a weak anomaly if

$$d_i \geq \mu + \tau_{out} \cdot \sigma.$$

We also determine a length threshold l. For a part of the polygonal chain, to be recognised as an outlier, it must hold that at least l successive points $ap_i, \ldots, ap_{i+l-1}$ must be strong anomalies. An outlier can have any length that is equal to or bigger than l. Finally, if we find such an outlier, we expand it by adding points to it that are adjacent to the outlier and weak anomalies. By this, it is possible for outliers to merge.

As an example, say, $ap_7, \ldots, ap_{20}$ are strong anomalies because their distances $d_7, \ldots, d_{20}$ are bigger than $\mu + \tau_{in} \cdot \sigma$. If we set $l = 4$ for this example, these thirteen points are recognised as an outlier $o = \{7, \ldots, 20\}$. Now, let us assume that ap_5 and ap_6 are weak anomalies as well as ap_{22}. Then ap_6 and ap_5 belong to the extended outlier, because they are both adjacent to the outlier. On the other hand, ap_{22} does not belong to the extended outlier, because ap_{21} is not part of the outlier since it is not a weak anomaly.

Once all outliers have been detected, the next step is to remove them. Currently, we use a very simple approach for this where, in the original silhouette S, the overhanging part of the outlier is replaced by a straight line. This is based on the notion, that the reasons for most outliers in the silhouette are either trees or clouds. By just removing them, in many cases, we get results that resemble the original shape of the mountain in an easy way.

Let $o = \{i, \ldots, j\}$ with $i + l \leq j$ be an outlier. Then we draw a straight line from v_i to v_j. Now, while the distance between v_s, starting with $s = i + 1$, and that straight is smaller than a preset value d, we find the vertex v_s with the largest index, that is close enough to the original straight line. The same is done from the other end of the outlier so that we find a vertex v_e. This results in four vertices' indices $j \leq s < e \leq j$[1]. From this, the part between v_s and v_e is now finally replaced by a straight line between these two vertices.

By this, we obtain a cleaned silhouette that can be used in further steps for the silhouette matching.

3.3 Silhouette Matching

Once the silhouette is extracted and cleared of outliers, it is now time to match it to the reference data in order to determine the mountain that is shown by the picture. Currently, we have not implemented any method, yet. However, we will introduce some methods that we are going to test in the future.

The converted silhouettes $AP = (ap_1, \ldots, ap_n)$ from the previous section can be easily changed to be interpreted as time series by setting $AP' = (ap'_1, \ldots, ap'_n)$ where

$$ap'_i = (l'_i, v_i) = (\sum_{j=1}^{i} l_i, v_i)$$

for $ap_i = (l_i, v_i)$. By this conversion, every point ap'_i has the length of the polygonal chain until that point as its first component. Since $l_i > 0$ for all $i > 1$, the length component of AP' is strictly monotonic increasing, just as the time dimension in a time series. This is because we do not allow identical points in our silhouette. With this, it is possible to compare different silhouettes for similarity by using time series comparison techniques such as [6, 9]. These methods have the advantage of being rotation invariant which, in our case, means that the image does not have to have the same image section as our reference data.

Due to the great number of mountains and peaks that exist and on the notion, that every peak can be photographed from different angles, there are hundreds of thousands of silhouettes to match for each query image. Due to this it is clear, that even with a high-performance matching method a preselection is necessary.

There exist many methods suitable for this, such as the technique presented in [7] that uses the position of the sun and the time of day, at which the photo has been taken, in order to compute the approximate location on earth. However, this method has an average localisation error of about 100 km. Because of this, it is useful for a rough preselection, however not sufficient in our case.

Therefore, we aim to reuse the idea of our outlier detection method for matching. Instead of computing histograms of short sequences of the silhouette, in this case the histogram H_S over the whole silhouette AP is computed. This is then compared to the reference images' histograms $H_{R_i}, i \in \{1, \ldots, n_{ref}\}$ with n_{ref} the number of the reference image, which we, as discussed in section 3.2, have to compute anyway. The comparison between two histograms, with our distance function, is linear to the number of buckets in the histograms, or, since this number is fixed, constant. Now, let $d(H_S, H_{R_i})$ denote the distance between the histogram of the new silhouette S and the ith reference histogram. If $d(H_S, H_{R_i})$ is small, this does not necessarily mean, that the silhouettes are identical or even really similar, because the histogram representation does not preserve the order of the contained elements. On the other hand, if the distance

[1]In general, it is possible, that $s \geq e$. In this case, no substitution is performed.

between two silhouettes is large, we can say that those silhouettes are not similar.

Due to this, we plan to only use the time series comparison methods from the beginning of this section on those reference silhouettes, that yield small histogram distances. Further on, we will evaluate if the position determination method presented in [7] is able to boost the performance of our solution.

4. CONCLUSION AND FUTURE WORK

In this work we presented a new approach to motif recognition based on silhouettes on the example of mountains. Our method consists of three steps, of which the first two have already been implemented while we are working on the third step. First results show that we are able to extract silhouettes with relatively few errors from images and that our outlier detection step does indeed find meaningful anomalies. We have currently tested the first two steps of our approach as described in sections 3.1 and 3.2 on 18 images from the reference dataset. Results for the first of these images can be seen in figures 2 and 4. Of these 18 images, 17 result in good silhouettes, i.e. silhouettes with relatively few outliers of which most get marked and are correctable. The last image, though, does not get recognised correctly due to low contrast in terms of brightness. This is illustrated by figure 6. We do, however, notice this, because most of the silhouette gets marked as outlier.

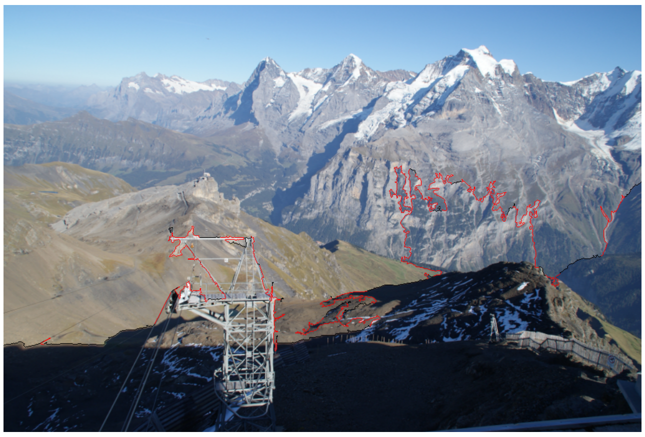

Figure 6: Silhouette as black line with outliers in red on original image.

The next step is to implement a silhouette matching algorithm. This has already been outlined in section 3.3. Of course, it is necessary, to benchmark the parts of that step in order to find weaknesses early on. Once the system is complete we aim to evaluate it on the whole of the dataset of [2]. Based on these result, we will tune the parameters of our method to find settings, that do work well generally.

We also aim to create a mountain recognition corpus of our own since [2] focuses on images of mountains in Switzerland only. Our corpus is aimed to be international and should feature images from mountains from all over the world.

Another interesting perspective is to test, whether our framework will work on other types of images, such as city skylines or pictures of single buildings. With these tasks the method itself could be quite similar, since skylines and buildings are mostly photographed with the sky as background, too. Furthermore, we aim to test our method on more diverse areas, such as the recognition of certain objects in MRT screenings or X-rays. These tasks will make it necessary to change some parts of our approach, naturally, because there, it will be interesting to be able to tell, for example, which organs are shown in a picture, or, as a next step, to be able to identify different kinds of tumours automatically.

5. REFERENCES

[1] Instagram @ONLINE, accessed April 7, 2015. `https://instagram.com/press/`.

[2] G. Baatz, O. Saurer, K. Köser, and M. Pollefeys. Large Scale Visual Geo-Localization of Images in Mountainous Terrain. In *Computer Vision - ECCV 2012*, Lecture Notes in Computer Science, pages 517–530. 2012.

[3] L. Baboud, M. Čadík, E. Eisemann, and H.-P. Seidel. Automatic Photo-to-terrain Alignment for the Annotation of Mountain Pictures. In *Proc. of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 41–48, 2011.

[4] R. Fedorov, P. Fraternali, and M. Tagliasacchi. Mountain Peak Identification in Visual Content Based on Coarse Digital Elevation Models. In *Proc. of the 3rd ACM International Workshop on Multimedia Analysis for Ecological Data*, MAED '14, pages 7–11, 2014.

[5] P. C. N. Jr, M. Mukunoki, M. Minoh, and K. Ikeda. Estimating Camera Position and Orientation from Geographical Map and Mountain Image. In *38th Research Meeting of the Pattern Sensing Group, Society of Instrument and Control Engineers*, pages 9–16, 1997.

[6] E. J. Keogh, L. Wei, X. Xi, S.-H. Lee, and M. Vlachos. LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures. In *VLDB*, pages 882–893, 2006.

[7] J.-F. Lalonde, S. G. Narasimhan, and A. A. Efros. What Do the Sun and the Sky Tell Us About the Camera? *International Journal of Computer Vision*, 88(1):24–51, 2010.

[8] 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.

[9] J. Lin, R. Khade, and Y. Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *J Intell Inf Syst*, 39(2):287–315, 2012.

[10] L. Porzi, S. R. Buló, P. Valigi, O. Lanz, and E. Ricci. Learning Contours for Automatic Annotations of Mountains Pictures on a Smartphone. In *Proc. of the International Conference on Distributed Smart Cameras*, ICDSC '14, pages 13:1–13:6, 2014.

[11] F. Y. Shih and S. Cheng. Automatic seeded region growing for color image segmentation. *Image and Vision Computing*, 23(10):877–886, 2005.

[12] L. Tao, L. Yuan, and J. Sun. SkyFinder: Attribute-based Sky Image Search. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 68:1–68:5, 2009.