SEMI-AUTOMATIC LANDFORM MAPPING AT STEEP SLOPES

Martin Schneider und Reinhard Klein

University of Bonn
Institute of Computer Science II
Römerstrasse 164, 53117 Bonn, Germany
ms,rk@cs.uni-bonn.de
http://www.cg.cs.uni-bonn.de

KEY WORDS: semi-automatic segmentation, landform mapping, image matching, surface parameterization

ABSTRACT:

The traditional mapping on digital aerial photos and elevation models has its limits at steep slopes where the surface is only very sparsely sampled even in high resolution data sets. As a consequence, there is too few information available to perform a detailed mapping. In this paper we present a framework for augmenting terrain data with additional photos and show how to utilize this additional information in the mapping process. The photos are matched to a textured elevation model by marking correspondence points through a simple point-and-click interface, which are then used to solve for the camera projection matrix. To compensate for different lighting conditions during acquisition, a histogram matching is applied in order to align the color distributions of the photo and the data set. The photos are rendered onto the terrain using projective texture mapping. If multiple overlapping photos are available, the best views are determined and blended together to avoid visible seams. The photos are incorporated into the segmentation process by computing a patch-wise, adaptive parameterization of the terrain geometry which defines a mapping of the surface into the plane with minimized distortion. The resulting image patches are connected appropriately at their boundaries and used as input for the segmentation algorithm.

1 INTRODUCTION

Landform mapping often serves as a basis for various kinds of further investigations of an area at focus. Maps compile knowledge on landforms, surface processes and surface materials that have a widespread application in land management practices, natural hazard assessments or landform evolution studies. Traditionally, mapping is based on field work supplemented by the interpretation of aerial photography and literature research. The availability and usage of high resolution digital elevation models, satellite and aerial images reduces the amount of time-consuming and costly field work. Modern GIS tools facilitate the compilation, production and distribution of maps by providing enhanced digitizing tools, data layers, data base functions, symbol creation, print and web publishing. However, most of the cartographic features of GIS software are limited to a two-dimensional representation of the base data. Although derivatives of the base data (slope-, aspect- and curvature maps) can be used to enhance the identification of objects, the natural three-dimensional human perception of the landscape is disabled by the restriction to a fixed viewpoint. In order to overcome these limitations semi-automatic landform mapping tools were presented in (Schneider and Klein, 2006, Schneider and Otto, 2006) that are integrated in a 3d visualization environment.

As a result of improved acquisition devices, aerial photography and digital elevation models with a resolution up to a few meters or even centimeters have become available. However, the resolution specification is with respect to a surface perpendicular to the acquisition direction, whereas the effective resolution of surfaces at oblique angles is much lower. Hence, the representation of steep slopes, which are of great interest in many disciplines, suffers from a sparse sampling. As a consequence, even in recent high resolution data sets there is usually insufficient information available in these areas to perform a detailed analysis or precise mapping. In this paper we present a mapping framework that gives the user the possibility to provide additional photos in order to accomplish a selective increase of resolution in areas of interest. The photos are matched to the textured elevation model

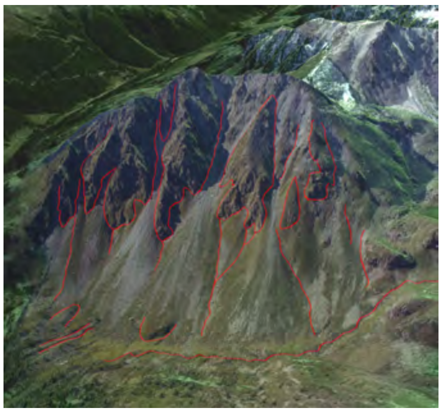
Figure 1: Segmentation performed on a textured elevation model with the help of additionally matched photos.

interactively by marking corresponding points in the photo and in the 3d environment, which allows solving for the camera matrix. The photos are visualized by means of projective texture mapping using the camera matrix resulting from the matching procedure. In order to prevent a projection of a photo through geometry and mapping it onto areas not visible from the calculated camera position, a shadow mapping algorithm is used to handle visibility. Since the terrain data and photos were probably acquired at different times and under different illumination conditions, a histogram matching algorithm is applied to the photos in order to adapt their color histograms to the data set's histogram. Additionally, determining the best available views for every surface point and blending them together helps to avoid visible seams between them.

The mapping algorithm, that originally only operates on the aerial photography and elevation data, is extended to incorporate the additional information supplied with the input photos (see Figure 1). For this purpose, we adaptively parameterize the terrain geometry on a per-patch basis. The parameterization defines a mapping of the surface to the plane optimized for area and, as much

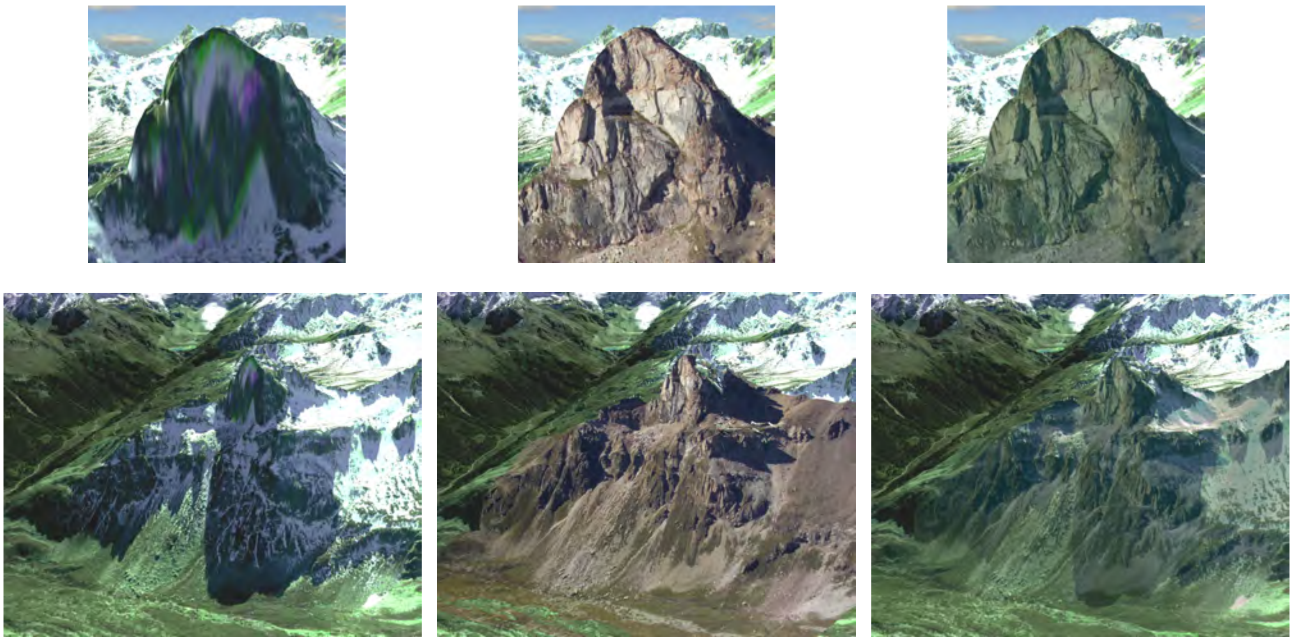
Figure 2: The bottom row shows a screenshot of the textured elevation model (left), with a photo mapped on it (middle) and with applied histogram matching and blending (right). The top row shows close-ups of the corresponding views below.

as possible, angle preservation, which results in a nearly uniform sampling of the surface. The segmentation algorithms can then be performed on these image patches. The patch boundaries have to be connected appropriately to assure smooth transitions between them.
The presented tools specifically target the mapping of geomorphological objects but are not limited to them. As a proof of concept we matched several photos to a HRSC (High Resolution Stereo Camera) data set of Turtmann valley (Switzerland) and used the revised mapping tools to map objects at steep slopes.

2 PREVIOUS WORK

Semi-automatic segmentation techniques provide high efficiency and accuracy by allowing users to do the object recognition and letting computers capture the fine details. These methods can basically be divided into region-based and boundary-based approaches.
Boundary-based methods cut out an object by allowing the user to surround its boundary with an evolving curve. The user traces along the object boundary and the system optimizes the curve in a piecewise manner. One well-known group of boundary-based techniques are those based on Intelligent Scissors (Mortenson and Barrett, 1995, Falcâo et al., 1998). Intelligent Scissors is a highly interactive tool which formulates the boundary detection problem in an image as a shortest path search in graph. While these tools provides highly interactive visual feedback once all shortest paths have been computed, it is time-consuming to recompute them especially when the image is large. Therefore, several attempts (Mortenson and Barrett, 1999, Falcâo et al., 2000, Wong et al., 2000, Kang and Shin, 2002) were presented that aim at increasing the efficiency of the boundary construction by restricting the search domain.
Region-based methods on the other hand work by allowing the user to give loose hints as to which parts of the image are foreground or background without the need to fully enclose regions. An underlying optimization algorithm extracts the actual object boundary based on the provided user input. In the seminal work (Boykov and Jolly, 2001) a graph cut optimization was used for this purpose. Since then, many approaches were published (Li et al., 2004, Rother et al., 2004) that extended and improved the original method, while others aimed at further accelerating the graph cut (Lombaert et al., 2005, Juan and Boykov, 2006).
In (Schneider and Klein, 2006, Schneider and Otto, 2006) two semi-automatic segmentation method were integrated into a 3d environment and applied to textured elevation data to allow landform mapping in 3d. Since these tools use out-of-core and hierarchical techniques they perform interactively even on very large data sets. Unfortunately, the methods do not allow a precise landform mapping at steep slopes due to the sparse sampling of the surface inherent in the aerial photography.

3 PHOTO MATCHING AND RENDERING

3.1 Matching

The task of matching a photo to a textured digital elevation model can be formulated as estimating the camera projection matrix which is known as resectioning. Given sufficiently many correspondences between world and image points the camera matrix can be determined from them by using the Gold Standard algorithm (Hartley and Zisserman, 2004). In our framework we let the user select appropriate correspondence points in the supplied photo and in the 3d the terrain through a simple point-and-click interface.
Given the point correspondences between 2d image points x_i and 3d world points X_i, a camera matrix P is estimated that maps the X_i onto the x_i, i.e. $PX_i = x_i$. Since P is a 3×4 matrix with 11 degrees of freedom, 11 equations are needed to solve for it. Because each point correspondence results in two equations at least 6 correspondences are needed. Given the minimum number of correspondences an exact solution can be found by organizing the equations in a matrix A and solving for $Ap = 0$, where p contains the entries of the camera matrix. If 6 or more points are provided, there is no exact solution and the maximum likelihood estimate of P is determined. To this end, the image points x_i as well as the world points X_i are normalized using similarity transformations T and U, respectively. Then, a linear solution is computed using the Direct Linear Transformation (DLT) algorithm which

computes the camera matrix as the unit singular vector of A corresponding to the smallest eigenvalue. This linear result is then used as input for the non-linear minimization of the geometric error $\sum_i \|x_i - PX_i\|^2$ with the Levenberg-Marquardt algorithm. Lastly, the camera matrix P of the original points is computed in a denormalization step from the estimated camera matrix $\overline{P}$ of the normalized points as $P = T^{-1}\overline{P}U$.

3.2 Rendering

Rendering the photographs on the terrain can be performed using projective texture mapping (Segal et al., 1992). Projective texture mapping is directly applicable to image-based rendering because it simulates the inverse projection of taking photographs with a camera and can be thought of as replacing the camera with a slide projector that projects the original photo back onto the terrain. In order to perform projective texture mapping, the estimated camera matrix is used to calculate the texture coordinates for each vertex by multiplying its 3d position with it. Since projective texture mapping does not automatically perform visibility tests, a photo will project through any geometry and be mapped onto polygons that are not visible from the camera's point of view. Thus, visibility information needs to be explicitly computed when using projective texture-mapping. While such occluded regions could be determined using object-space visible surface or image-space ray casting algorithms, we apply an image-space shadow map algorithm (Williams, 1978). To this end, we once render the scene from the camera's point of view in an offscreen buffer and store the depth values in a depth map. During rendering each fragment is transformed to the camera's coordinate system and its depth value is compared to the corresponding value in the depth map. If the depth value stored in the depth map is smaller than the depth value of the current fragment, it is not visible from the camera's point of view and therefore rejected.
In general, a photograph will only contain a part of the object of interest. Thus, it is usually necessary to combine multiple overlapping images in order to render the entire object at increased resolution. Consequently, some parts of the object are covered by only one while others might be covered by several photos. If a surface point is contained in multiple images, the renderer has to decide which image or combinations of them to use. What is more, the images will usually not agree perfectly in the overlapping areas due to different lighting conditions, non-lambertian reflection or unmodeled geometric detail in the terrain surface.

3.3 Histogram matching

To account for the different color distributions in the data a histogram matching is applied, which adapts the distribution of color values in the photo to that of the aerial photography. We use the histogram matching proposed in (Reinhard et al., 2001) that transfers the color characteristics from a source to a target image. In our case the target images are the input photos. The corresponding source images are created by rendering the scene from the previously computed camera's point of view corresponding to that photo.
In this algorithm the rgb values of the source and target image are first transformed into the orthogonal $l\alpha\beta$ color space. Since the axes in the $l\alpha\beta$ space are decorrelated, different operations can be performed on the different color channels independently without introducing cross-channel artifacts. Since we want to transfer some properties of the distribution of color values from the source to the target image, mean μ_s, μ_t and standard deviation σ_s, σ_t are computed for both. Then, each color value in the target image is

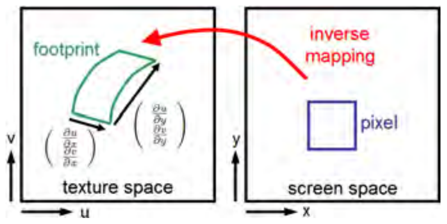
footprint
inverse mapping
pixel
$\begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial x} \end{pmatrix}$
$\begin{pmatrix} \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial y} \end{pmatrix}$
v
u
y
x
texture space
screen space

Figure 3: A square screen pixel that covers a curved surface has a curvilinear quadrilateral preimage in texture space called footprint. This footprint can be approximated by a parallelogram.

transformed as follows

$$\begin{aligned} l' &= \frac{l - \mu_t^l}{\sigma_t^l}\sigma_s^l + \mu_s^l \\ \alpha' &= \frac{\alpha - \mu_t^\alpha}{\sigma_t^\alpha}\sigma_s^\alpha + \mu_s^\alpha \\ \beta' &= \frac{\beta - \mu_t^\beta}{\sigma_t^\beta}\sigma_s^\beta + \mu_s^\beta . \end{aligned}$$

After this transformation the resulting color values in the target image have a mean and standard deviation conform to that of the source image. Finally, the color values are converted back to rgb. The quality of the result of the algorithm depends on the similarity in composition of the source and target image. Since in our case source and target image show the same scene from the same viewpoint, they are usually identical in composition and the algorithm produces satisfying results.

3.4 Blending

Although histogram matching helps to adapt the photos to the data set and each other, they still do not match perfectly in the overlapping regions and at the borders. Using only a single view at every pixel means that neighboring pixels may be sampled from different original photos, which can cause visible seams in a rendering. To account for this, transitions are smoothed by performing a weighted averaging of the best available views. Computing the corresponding weighting factors based on the angle between surface normal and acquisition direction alone, completely ignores the resolution of the photos. Instead, we compute for each pixel a blending factor w_f by estimating the area of the footprint of this pixel (see Figure 3). The footprint is the preimage of the pixel in the photo and its area is thus a measure of how much information the photo contains about this part of the surface. The footprint can be approximated by a parallelogram spanned by the vectors

$$f_x = \begin{pmatrix} \frac{\partial u}{\partial x} \\ \frac{\partial v}{\partial x} \end{pmatrix} \text{ and } f_y = \begin{pmatrix} \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial y} \end{pmatrix},$$

consisting of the partial derivatives of the texture coordinates u and v in x and y direction. The area of the footprint is then computed as $w_f = |f_x \times f_y|$. Even with this weighting, neighboring pixels can still be sampled from different views at the boundaries of a projected image, because the contribution of an image has to be zero outside its boundary. To address this, pixel boundary weights w_b are introduced that are 1 in the image's center and fall off towards the border. The final weight b for an image's pixel is then computed as $w = \min(w_f, w_b)\, v$, where v is a visibility term that is 0 if the fragment is not visible from the camera's point of view and 1 otherwise. The final color is then computed as a weighted sum of the pixels with the largest blending factors w. Although this method does not guarantee smooth transitions in all cases, it proved to eliminate most artifacts during rendering in practice.

4 REVIEW OF PREVIOUS SEGMENTATION METHODS

In this section we will shortly review the preliminary work (Schneider and Klein, 2006, Schneider and Otto, 2006) the presented framework is build upon. In that work traditional mapping tools were extended by integrating semi-automatic segmentation tools in a 3d visualization environment. This combination allows a real-time exploration of the terrain and marking of objects of interest at the same time. By navigating in the 3d environment, landforms can be inspected from arbitrary views, which helps the user to identify objects of interest. The user is able to perform the mapping of an object in this 3d environment by simply marking it directly on the terrain surface. The tools simplify the mapping of objects because they demand only loose mouse gestures as input, indicating the approximate position of the object. In contrast to common manual mapping, which is tedious and time consuming, this results in reduced user input without sacrificing accuracy. The two tools presented are a boundary-based approach based on Intelligent Scissors and a region-based approach using a graph cut optimization. Both were optimized to work on a quadtree representation of the data. The base level of the quadtree contains the original data partitioned into equally sized square tiles. Tiles on higher levels of the hierarchy are created by merging 2×2 from the next lower level and downsampling them by a factor of 2. The quadtree data structure allows to efficiently handle out-of-core data sets since it provides fast access to spatial subparts of the data set at different levels of detail.

4.1 Intelligent Scissors based segmentation

The Intelligent Scissors based segmentation algorithm formulates the boundary detection problem as an optimal path search in a graph. The objective is to find the optimal path from a seed node to a destination node where pixels in the image represent nodes with directed and weighted edges connecting its eight adjacent neighbours. An optimal path is defined by the minimum cost path, i.e. a path with the smallest sum of edge costs. Since a shortest path in the graph should correspond to an object boundary in the image, pixels with strong edge features in the image should lead to low local costs in the graph and vice-versa. Hence, local costs are created as a weighted sum of the edge features.
Despite the fact that the Intelligent Scissors technique provides a powerful tool for image segmentation its speed and memory consumption constrain its feasibility when large data sets need to be processed. Since in the case of terrain data usually very large data sets must be processed, the direct application of the original Intelligent Scissors approach is not possible. In order to ensure interactive response even on very large data sets the quadtree data structure is exploited in two ways: Localizing the search domain within a quadtree level and employing a multilevel banded heuristic to exploit the hierarchical structure. Localizing the search domain means to search for a shortest path only in a restricted area around the user input. The search domain is incrementally extended based on the user input and necessary parts of the data set are loaded from disk and corresponding edge features are computed on-the-fly. As a result only a very small subset of the whole data set has to be held in memory while the majority resides on disk. The multilevel banded heuristic starts the segmentation on a coarser level and propagates the result to the next higher resolution level where the segmentation is performed only within a narrow band surrounding the projected result from the coarser level. This procedure is repeated until the highest resolution level is reached.

4.2 Graph cut based segmentation

The graph cut based algorithm formulates object extraction as a binary labeling problem which assigns to each node p in a graph, i.e. pixel, a unique label $x_p \in \{\text{foreground, background}\}$. The solution $X = \{x_p\}$ can be obtained by minimizing a Gibbs energy (Geman and Geman, 1984)

$$G(X) = \sum_p R(x_p) + \lambda \sum_{(p,q)} B(x_p, x_q).$$

$R(x_p)$ is the likelihood energy that encodes the cost when the label of node p is x_p, i.e. it encodes the likelihood that the pixel belongs to the foreground or background. The likelihood is estimated as the color similarity of the pixel's color to the color distribution of the areas marked by the user as foreground and background. The prior energy $B(x_p, x_q)$ denotes the cost when the labels of adjacent nodes p and q are x_p and x_q, respectively. B is defined as a function of the color gradient between the two nodes. In other words, B is a penalty term when adjacent nodes are assigned with different labels. The more similar the colors of the two nodes are, the larger is B, and thus the less likely the edge is on the object boundary. The influence of the region and boundary terms is controlled by the weighting factor λ. Decreasing λ leads to more complex and longer boundaries and generally increases the number of resulting objects, especially small ones. The energy function $G(X)$ is minimized using the max-flow algorithm presented in (Boykov and Kolmogorov, 2001) which is especially designed for vision problems.
Since it is crucial to generate the object boundary with very little delay, a multilevel banded heuristic (Lombaert et al., 2005) is applied to reduce both running time and memory consumption of the graph cut optimization. The procedure is similar to the heuristic proposed to accelerate the Intelligent Scissors based approach.

5 REVISED SEGMENTATION ALGORITHM

A limitation of all segmentation tools working solely on aerial photography and elevation data is that they cannot be used to perform a reasonable mapping at steep slopes due to lack of data in these areas. Therefore, we revise our segmentation algorithms to

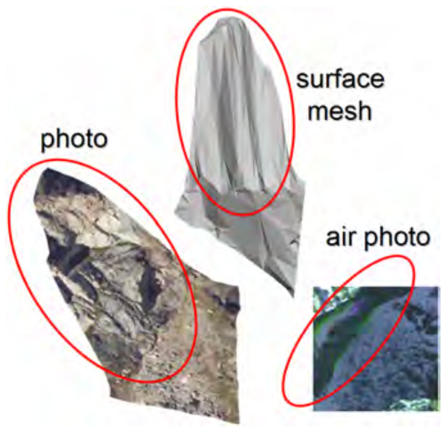

Figure 4: An example of a surface geometry patch (top). The corresponding textured projections into the plane using the computed parameterization and a photo (bottom left) and using orthogonal projection and the aerial photography (bottom right). Corresponding areas are marked by red ellipses demonstrating the irregular sampling of the surface in the air photo.

be able to incorporate the information contained in the matched photos in addition to the aerial photographs. To this end, we have to find a suitable mapping of the surface into the plane. For an infinitesimal small surface patch a projection to its tangential plane defines a perfect mapping. In contrast to that, parameterizing the terrain surface as a whole introduces noticeable distortions. As a reasonable compromise we apply an adaptive, i.e. distortion-controlled, parameterization of the terrain surface in a preprocessing step. For this purpose, we start on the finest level of the quadtree and parameterize each patch in it. Then, we recursively merge and parameterize neighboring 2×2 tiles until the the distortion imposed by the parameterization exceeds a given threshold.

5.1 Parameterization background

A parameterization of a surface can be viewed as a one-to-one mapping from a suitable domain to the surface. These surfaces are usually represented by triangular meshes and the mappings are linear over the triangles. Parameterizations almost always introduce distortion in either angles or areas. Most applications demand parameterizations that minimize these distortions in some sense. Many different ways of achieving this have been proposed in the literature. A comprehensive survey of local parameterization methods can be found in (Floater and Hormann, 2005).
Given an orientable 2-manifold surface patch $S \subset \mathbb{R}^k$ a parameterization is defined as a homeomorphism

$$\begin{aligned} \phi : \Omega \subset \mathbb{R}^2 &\rightarrow S \\ (u,v) &\mapsto \phi(u,v) \end{aligned}$$

from the parameter space Ω into S. In the following we consider the problem of finding a parameterization for a set S that has a triangulation $M_S = \{[1 \ldots n], T, (p_i)_{i=1\ldots n}\}$. Furthermore, we require the inverse parameterization ϕ^{-1} to be linear within the triangles of M_S. Such a mapping is uniquely determined by its values $((u_i, v_i))_{i=1\ldots n} := (\phi^{-1}(p_i))_{i=1\ldots n}$ on the mesh vertices and $M_\Omega = \{[1 \ldots n], T, ((u_i, v_i))_{i=1\ldots n}\}$ is a parameter domain triangulation for the image $\phi^{-1}(S)$. The inverse parameterization ϕ^{-1} maps vertices and faces of M_S onto vertices and faces of M_Ω respectively.

5.2 Parameterization of geometry patches

Many parameterization methods demand the boundary mapping to be fixed in advance and map to convex polygons which may be sufficient or even desirable for some applications. In our case it would be advantageous because fixing the boundary vertices and mapping to a square would result in patches with coinciding borders. However, such restrictions usually introduce additional distortion. For the sake of quality we decided to use a parameterization method that does not constrain the boundary. This, however, comes at the cost of having to handle patch borders appropriately, since they do not match.
We use the parameterization presented in (Degener et al., 2003) to parameterize each surface geometry patch in the quadtree representing the data set. The algorithm quantifies angle and global area deformations simultaneously and lets the user control the relative importance between them through a parameter. We choose this parameter in order to obtain a parameterization that is optimized for a uniform sampling of the surface.
A 2d image of the patch can then be created by rendering the geometry of the patch where its 2d texture coordinates are used as vertex positions (see Figure 4). The corresponding texture coordinates for accessing the photo are computed by multiplying the original vertex position with the camera matrix. Since the surface geometry patches in our data set contain usually about up to a thousand vertices, parameterizing a patch with the aforementioned algorithm might take up to about several seconds. Because of this, we perform the parameterization in a preprocessing step, store the texture coordinates on disk and load them on demand.

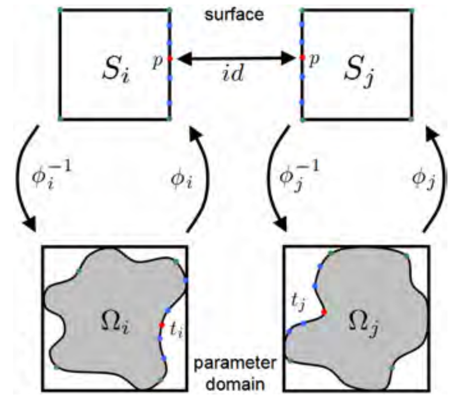

Figure 5: Surface patches S (top) and their respective domains Ω (bottom). Projecting the surface into the domain results in arbitrarily shaped patches (grey). Note that neighboring patches have common borders, which allows transitions from one patch to another.

5.3 Handling patch borders

The border of neighboring patches that were not parameterized together do not have the same borders in the parameter domain and therefore demand a special handling. Given a point t_i in the parameter domain Ω_i at the border of patch i, we want to find its neightbor t_j in the parameter domain Ω_j of the adjacent patch j (see Figure 5). Since t_i is a border vertex it is located on an edge and can therefore be represented as the linear combination of its edge vertices. Assuming the edge vertices are $t_{i,0}$ and $t_{i,1}$ then t_i can be expressed as

$$t_i = \lambda_i t_{i,0} + (1 - \lambda_i) t_{i,1}.$$

Since the parameterization is linear over the triangles, we can transform it into S_i as

$$\begin{aligned} \phi_i(t_i) &= \lambda_i \phi_i(t_{i,0}) + (1 - \lambda_i)\phi_i(t_{i,1}) \\ &= \lambda_i p_{i,0} + (1 - \lambda_i) p_{i,1} = p. \end{aligned}$$

Since neighboring geometry patches have common boundaries we can write p as a linear combination of vertices in S_j

$$p = \lambda_j p_{j,0} + (1 - \lambda_j) p_{j,1}$$

Again using the linearity condition we can transform the result to the parameter domain by applying ϕ_j^{-1}. Then, the corresponding point t_j in Ω_j can be computed as

$$t_j = \lambda_j \phi_j^{-1}(p_{j,0}) + (1 - \lambda_j)\phi_j^{-1}(p_{j,1}).$$

However, in the discrete case a pixel in the texture represents an interval in the domain Ω. That means that we map an interval on the border of Ω_i to an interval on the border of Ω_j which must not necessarily have the same length. Consequently, we have no longer a one-to-one mapping of border pixels but a one-to-many mapping. This does not affect the segmentation algorithms, since they do not demand a fixed number of neighbors per node or a grid graph. The mapping functions ϕ and ϕ^{-1} can be efficiently implemented as a lookup table by rendering the geometry color coded with texture coordinates as vertex coordinates and vertex coordinates as colors for ϕ and vice-versa for ϕ^{-1}.

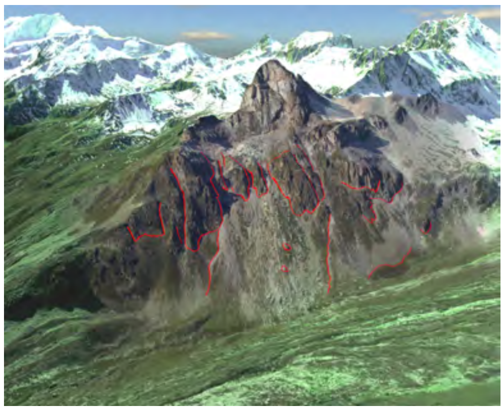

Figure 6: Segmentation result performed on a matched photo.

For both algorithms image features, like image gradients, have to be computed that serve as edge costs in the graph. Applying filter kernels to the created image patches to compute them leads to biased results at the borders due to missing information from neighboring patches. To this end, we parameterize geometry patches that slightly overlap, so that the overlap is at least as big as half the size of the used filter kernel. Proceeding this way the introduction of artifical edges at the borders is avoided.

6 RESULTS AND CONCLUSION

To assess the usability of our system, we first reparameterized and then mapped several photos to an HRSC-A data set of Turtmann valley in Switzerland. The data set covers an area of about 200 km^2 and has a resolution of 1m for aerial photography as well as elevation data. Unfortunately, the HRSC-A data does only contain a red channel that is shifted to near-infrared, which is why the data set's color look, despite postprocessing, somewhat unnatural. An additional difficulty is that the used photos were acquired during summer when snow was only present at the highest mountains, while the data set was generated late in the year with snow also in lower-lying areas. This presents a challenge for the histogram matching and blending because images acquired under completely different conditions have to be combined. Nevertheless, the results look convincing in most parts (see Figure 2).
Augmenting a textured elevation model with additional photos drastically enhances its visual quality and information content. Especially the in the aerial photography only sparsely sampled steep slopes benefit from the additional information and their resolution can be increased by orders of magnitude, which is a requirement for a detailed landform mapping in these areas. Figure 1 and 6 show segmentation results at steep slopes using information from matched photos. Moreover, photos can not only increase resolution but contain information of parts of the surface that might be hidden or not clearly visible in the aerial photography due to snow or shadows, for example. Additionally, other segmentation methods working on aerial photography and elevation data can benefit from the proposed parameterization as long as they can handle arbitrarily shaped image patches.

REFERENCES

Boykov, Y. and Jolly, M.-P., 2001. Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. Proceedings of ICCV 1, pp. 105–112.

Boykov, Y. and Kolmogorov, V., 2001. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. Energy Minimization Methods in Computer Vision and Pattern Recognition pp. 359–374.

Degener, P., Meseth, J. and Klein, R., 2003. An adaptable surface parametrization method. The 12th International Meshing Roundtable.

Falcâo, A. X., Udupa., J. K. and Miyazawa, F. K., 2000. An ultrafast user-steered image segmentation paradigm: live wire on the fly. IEEE Transactions on Medical Imaging 19(1), pp. 55–62.

Falcâo, A. X., Udupa., J. K., Samarasekera, S., Sharma, S., E., B. and Lotufo, R., 1998. User-steered image segmentation paradigms: live wire and live lane. Graphical Models and Image Processing 60, pp. 233–260.

Floater, M. S. and Hormann, K., 2005. Surface parameterization: a tutorial and survey. Advances in Multiresolution for Geometric Modelling pp. 157–186.

Geman, S. and Geman, D., 1984. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 6, pp. 721–741.

Hartley, R. I. and Zisserman, A., 2004. Multiple View Geometry in Computer Vision. Second edn, Cambridge University Press, ISBN: 0521540518.

Juan, O. and Boykov, Y., 2006. Active graph cuts. IEEE conference on Computer Vision and Pattern Recognition.

Kang, H. W. and Shin, S. Y., 2002. Enhanced lane: interactive image segmentation by incremental path map construction. Graphical Models 64(5), pp. 292–303.

Li, Y., Sun, J., Tang, C.-K. and Shum, H.-Y., 2004. Lazy snapping. ACM Transaction on Graphics.

Lombaert, H., Sun, Y., Grady, L. and Xu, C., 2005. A multilevel banded graph cuts method for fast image segmentation. Proceedings of ICCV pp. 259–265.

Mortenson, E. N. and Barrett, W. A., 1995. Intelligent scissors for image composition. SIGGRAPH 95 Proceedings pp. 191–198.

Mortenson, E. N. and Barrett, W. A., 1999. Toboggan-based intelligent scissors with a four parameter edge model. Proceedings of IEEE Computer Vision and Pattern Recognition 2, pp. 452–458.

Reinhard, E., Ashikhmin, M., Gooch, B. and Shirley, P., 2001. Color transfer between images. IEEE Computer Graphics and Applications 21(5), pp. 34–41.

Rother, C., Kolmogorov, V. and Blake, A., 2004. Grabcut - interactive foreground extraction using iterated graph cuts. ACM Transaction On Graphics 23(3), pp. 309–314.

Schneider, M. and Klein, R., 2006. A multilevel banded intelligent scissors method for fast segmentation in large virtual terrains. II International Conference Remote Sensing Archaeology.

Schneider, M. and Otto, J.-C., 2006. A new semi-automatic tool for 3d landform mapping. 9th International Symposium on High Mountain Remote Sensing Cartography.

Segal, M., Korobkin, C., van Widenfelt, R., Foran, J. and Haeberli, P., 1992. Fast shadows and lighting effects using texture mapping. Proceedings of the 19th annual conference on Computer graphics and interactive techniques pp. 249–252.

Williams, L., 1978. Casting curved shadows on curved surfaces. In SIGGRAPH 78 pp. 270–274.

Wong, K. C., Heng, P. A. and Wong, T. T., 2000. Accelerating intelligent scissors using slimmed graphs. Journal of Graphic Tools 5(2), pp. 1–13.