Enhancing Textured Digital Elevation Models Using Photographs

Martin Schneider and Reinhard Klein
Institute of Computer Science II
University of Bonn
Römerstrasse 164, 53117 Bonn
{ms,rk}@cs.uni-bonn.de

Abstract

We present a method for enhancing the visual quality of existing digital elevation models textured with orthophotos, using a sparse set of unordered, high resolution photographs. After an initial manual selection of correspondence points, we automatically register the input photographs to the given terrain data set using robust image-based modeling techniques. To combine the geo-registered images on the terrain surface, we propose a compositing algorithm that ensures smooth transitions between the images while at the same time preserving the fine details. The resulting textures are inserted into the quadtree representation of a terrain rendering engine to allow an efficient real-time visualization. We demonstrate our method on an HRSC terrain data set and a collection of high resolution photos of Turtmann valley in Switzerland.

1 Introduction

Since their origin in the late 1950s, digital elevation models (DEMs) along with the corresponding aerial imagery have found wide application in various disciplines such as mapping, remote sensing, land planning and communications. During this same time, acquisition and processing of terrain data as well as corresponding visualization techniques have continuously progressed so that nowadays a real-time visualization of large, high-resolution terrain data sets has become feasible. However, a major drawback of aerial imagery is the irregular sampling of the terrain surface leading to a coarse representation of steep slopes while overhangs are not captured at all. In addition to poor visual quality, this severely limits the applicability of the data in disciplines such as geology or geomorphology.

Considering the quality and availability of today's digital cameras, the use of high resolution photos offers an easy and affordable way to capture additional information in areas of interest. Concerning the necessary fusion of the different data two main challenges arise: registration and compositing. Registration involves the recovering of the photos' camera parameters in order to place them into a common 3D coordinate system with respect to the DEM. The essential problems in compositing the registered images include the choice of a parameterization of the compositing surface, a selection of pixels that contribute to the final image and a blending of these pixels to minimize visible seams, blur and ghosting.

We present in this paper a method for enhancing the visual quality of existing digital elevation models textured with orthophotos, using a sparse set of unordered, high resolution photographs. As particular contributions, our paper presents:

- an extension of state-of-the-art structure from motion procedures by integrating the given textured DEM into the optimization,
- a fast color matching algorithm between multiple, overlapping images to remove large scale color and lightness shifts,
- a high-quality patchwise blending approach in the texture domain using a local reparameterization of the terrain surface.

Figure 1 provides a high level overview of the processing pipeline. As a preprocessing step, we reparameterize steep areas of the terrain surface in order to obtain a reasonable representation for the final textures. In the registration process we extract distinctive features from each image and match them to establish global correspondences. We utilize an interactive technique to estimate an initial georeferenced camera and use it to initialize an incremental structure from motion (SfM) procedure in which the remaining cameras are added one by one. Once the images have been registered they are combined to textures for the terrain in a compositing step. Compositing starts by determining visibility for each view in order to identify the

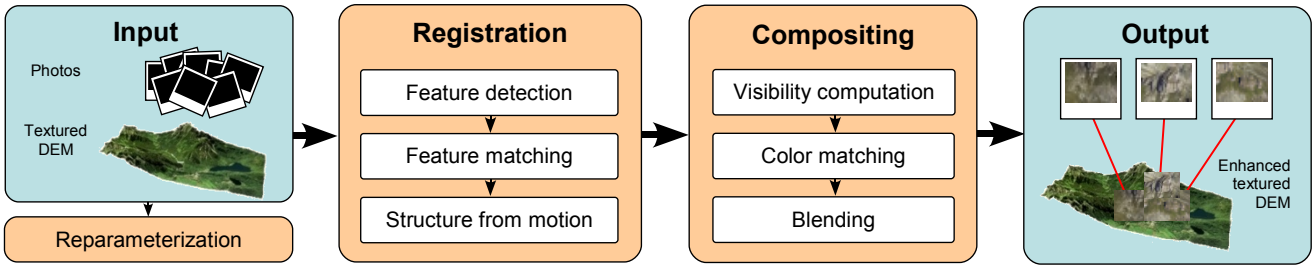

Figure 1. Overview of our approach to enhance ortho-textured DEMs using photographs.

regions on the terrain surface valid for texturing with the respective image. After that, the registered photos are merged in a two-step procedure. First, the color distributions in the images are matched in order to remove large scale color and lightness shifts. In the second step, the final textures are created by applying a weighted pyramid blending technique in texture space induced by the local reparameterization of the terrain geometry. Finally, the blended images are inserted into the quadtree data structure of the terrain rendering engine in order to allow a real-time visualization.
The remainder of this paper is organized as follows: In Section 2 we briefly review the relevant literature related to our work followed by a description of the reparameterization of the terrain in Section 3. We present our approach to georegister the photos in Section 4 and describe the compositing of the images to textures in Section 5. In Section 6 we present results, discuss limitations and future work, and conclude in Section 7.

2 Related Work

There are two main categories of work related to ours: image-based modeling and image-based rendering.

2.1 Image-based modeling

Image-based modeling is the process of creating three-dimensional models from a set of input images [7][22]. Our image-based modeling approach is based on recent work in structure from motion, which aims to recover camera parameters, pose estimates and sparse 3D scene geometry from image sequences.
In [25], Schaffalitzky et al. presented a method for estimating camera parameters from unordered image sets with the main focus on an efficient matching of feature points between photos. Robust techniques for multiview reconstruction given pairwise euclidean reconstructions were presented in [16][17]. Our SfM approach is based on the iterative methods presented in [5][26] in which cameras are added to a bundle adjustment one by one to increase robustness. In contrast to previous approaches that used much smaller and simpler data sets, the work by Snavely et al. marked the first successful demonstration of SfM techniques applied to large, real-world image sets. To align the cameras to a DEM they proposed the application of an interactive technique after the reconstruction followed by a re-run of the SfM.
Unlike them, we align an initial camera to the DEM in advance, which allows us to make use of the textured DEM during the SfM procedure. In contrast to the aforementioned work our main objective is not to estimate 3D geometry as best as possible but to fit the photos as best as possible to the existing approximate terrain geometry.

2.2 Image-based rendering

In image-based rendering new views of a scene are synthesized from a set of input images. Creating image-based texture maps for a 3D object is essentially the problem of combining texture fragments. A common approach is to apply triangle-based mosaicing schemes and use feathering to mask seams afterwards [19][13][24]. In general these techniques rely on a regular triangular mesh model and each triangle is assigned to the best camera by considering viewing angle and visibility.
An alternative to mosaicing schemes is to use per-pixel weighted filtering over the whole mesh [2][3][20][21]. Smooth weight functions are used to assure continuous transitions and to avoid visible seams. Similar to Baumberg [2], we also use a weighted multi-band blending approach to build view-independent textures from photos. By contrast, we perform a patchwise reparameterization of the 3D geometry and apply the blending in texture space.

3 Reparameterization

We use the terrain rendering system presented in [29] for the visualization of the textured DEM. It is based on a quadtree representation of the data where each quadtree tile contains corresponding geometry and texture data. The goal

of our method is to create enhanced textures using the input photos and to store them in the quadtree. Typically, ortho-projection is used to parameterize terrain geometry which is reasonable as long as only aerial imagery is involved or if the terrain is flat, otherwise large distortions are introduced. To obtain an appropriate representation for textures created from images from arbitrary views, we perform a local reparameterization of steep slopes in a preprocessing step. In order to identify the tiles that need to be reparameterized, we use a simple thresholding scheme. We compute the area of the tile's geometry and divide it by the area of its ortho-projection. If this ratio exceeds a given threshold, we reparameterize the tile.

We use the algorithm presented in [8] to parameterize the terrain geometry. It quantifies angle and global area deformations simultaneously and lets the user control the relative importance. We choose the importance in order to obtain a parameterization that is optimized for a uniform sampling of the surface. As the reparameterization is performed locally, i.e. for each tile separately, the introduced distortion is very low. Since the resulting texture space representation of a geometry tile has arbitrarily shaped boundaries and arbitrary orientation, we optimize it for later storage in a texture map. To this end, we compute its minimum area bounding rectangle using rotating calipers [27] and rotate it in such a way that its bounding box becomes axis aligned.

4 Registration

Our registration approach is based on the method presented by Snavely et al. in [26]. In their approach they perform a feature detection and matching, followed by an incremental SfM optimization. The resulting relative camera parameters can optionally be geo-referenced after the SfM optimization using an interactive technique. In contrast to them, we geo-register an initial camera before running the SfM and use it as starting point for the optimization. The benefit of doing this is that we can exploit the information contained in the textured DEM during the SfM optimization.

4.1 Structure from motion

We use the SIFT keypoint detector [14] to find feature points in each of the input photos and then perform a pairwise matching of features between each image pair. For outlier removal we estimate a fundamental matrix based on the detected feature points inside a RANSAC [9] procedure. Feature points that are not compatible with the computed epipolar geometry are removed from the optimization. The remaining features are arranged into connected sets of matching keypoints across multiple images. Given a set of connected features, we use bundle adjustment to solve

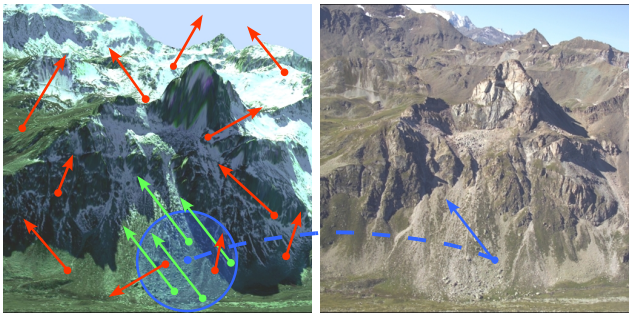

Figure 2. Feature matching between a photo (right) and the corresponding rendering (left). A feature in the photo is only compared to a small subset (green) of the rendering's features that roughly coincide with respect to position, scale and orientation.

for the camera parameters. Since SfM problems are prone to bad local minima they require good initial estimates. To account for this, cameras are added one by one into the optimization starting with the camera that contains the largest number of matches.

In order to be able to exploit the DEM in the following SfM optimization we geo-reference the initial camera in advance. For this purpose, we let the user specify a few correspondences between the photo and the textured DEM and then estimate the camera parameters using the direct linear transform (DLT) method [11] inside a RANSAC procedure. We utilize the textured DEM in two ways during the SfM optimization: first, we use it to obtain 3D estimates for the features and, second, to create correspondences between the input photos and the aerial imagery. To this end, we render the textured DEM from the camera's point of view with the resolution of the corresponding photo. The 3D position of all features in the image can then be initialized using the respective depth buffer value. Thus, by using the DEM we always have a 3D estimate for all features in an image available as soon as the respective camera is estimated.

To establish correspondences between the photo and the aerial imagery we extract feature points from the rendered image and match them with the features detected in the photo. Depending on the quality of the aerial imagery it is usually not possible to detect many reliable matches. However, in contrast to the feature matching between photos performed previously, we now have additional information at hand, namely an estimate of the camera parameters. Consequently, the photo and the rendered image are already roughly aligned which we can use to reduce the set of potential matching candidates significantly and hence increase robustness of the matching. Given a feature at position x in

the photo, we do not compare it to all features found in the rendered image but only to those features that are within a certain distance to x in the rendering and also have a similar scale and orientation (see Figure 2). If a reasonable number of consistent matches is found they are added as ground control points to the optimization. Although not every image can be registered directly to the terrain in this manner the remaining images are indirectly constrained by them. As a consequence potential drift is reduced and robustness of the optimization is increased.
When a new camera is added to the optimization we initialize all its 2D features with 3D points transferred from matching features in other images that already have a 3D estimate. Then, we use the DLT technique to estimate the added camera. Once the camera is estimated, we assign to the still uninitialized 2D feature points corresponding 3D points obtained from the DEM and try to add new correspondences between the photo and the aerial imagery. Outliers are removed after each run considering their reprojection error and 3D estimate. This procedure is repeated until no remaining camera has any matches with features already in the optimization.

5 Compositing

Once the images have been registered they are combined to textures for the terrain in a compositing step. Compositing starts by determining visibility for each view in order to identify the regions on the terrain surface valid for texturing with the respective image. With this information at hand, the registered photos are combined in a two-step procedure. First, color distributions in the images are adapted in order to remove large scale color and lightness shifts. In the second step, the final textures are created by applying a weighted pyramid blending approach in texture space induced by the local reparameterization of the terrain geometry.

5.1 Visibility Computation

In order to texture the terrain geometry with an image from a certain view, it is necessary to identify the visible parts of the surface with respect to this viewpoint and restrict texturing accordingly. We use a simple image space algorithm to determine visibility for each camera separately. The output of the visibility computation for a certain view consists of a list of the completely visible triangles and the visible subpolygons of the partially visible triangles.
Given a camera, we assign each triangle a unique color and render the terrain geometry from the camera's viewpoint into an offscreen buffer. The rendered image is then traversed to identify all rasterized triangles by their color ID.

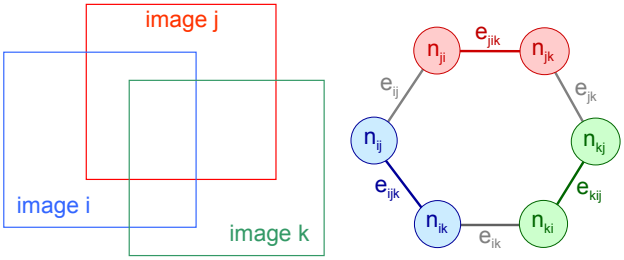

Figure 3. Simple example of three overlapping images and the corresponding graph.

In addition to that, we count the number of rendered pixels for each triangle and find its neighbors in image space. In the next step, we determine if the gathered triangles are completely visible or only partially visible. To this end, we render the triangles in question again but this time with depth test disabled so that they are completely rasterized and use occlusion queries to count the number of pixels. If the number of rendered pixels of a triangle in the two passes coincide, it is completely visible otherwise only partially visible. Finally, we calculate the visible areas of the partially visible triangles analytically in image space. We clip each triangle against its image space neighbors that lie in front of it and against the viewport. The resulting 2D polygons are unprojected to obtain the visible subpolygons of the triangle in object space.
Note that even at high framebuffer resolutions it cannot be ruled out that very small but visible triangles do not result in a rendered pixel due to discretization. However, we did not encounter such cases in practice that would have made any special case handling necessary.

5.2 Color Matching

When texturing a 3D object with photos taken from different viewing angles and with different camera settings, measured color and intensity values of a surface element observed in the different photos do usually not agree. The reasons for this are various and include, for example, view-dependent lighting effects, such as highlights and specularities, and variations in the camera gain settings and lead to a mosaic appearance on the surface when combining the images. To reduce color differences, methods that perform a color matching between image pairs can be used. However, applying pairwise color correction to multiple overlapping images is difficult considering the potentially complex topology of overlaps and usually does not result in a globally optimal solution. To overcome this problem, Bannai et al. presented in [1] a simultaneous color matching of multiple overlapping images based on minimizing per-pixel

differences in the overlapping regions. Unfortunately, minimizing per-pixel differences requires a precise registration and involves high computational costs.
In contrast to that, we extend the pairwise color matching approach by Reinhard et al. [23] to a simultaneous matching of multiple overlapping images. In the original approach they transferred color characteristics from a source to a target image in $l\alpha\beta$ color space. The new color ${p'}_t(x)$ in the target image is computed from the old color $p_t(x)$ as

$$ {p'}_t(x) = \frac{p_t(x) - \mu_t}{\sigma_t}\sigma_s + \mu_s, $$

where μ_s, μ_t are the means and σ_s, σ_t the standard deviations of the underlying gaussian distribution in the $l\alpha\beta$ color space of the respective source and target images. Using color distributions instead of per-pixel differences allows us to handle misregistrations robustly and to reduce computational costs significantly. In order to define the objective function for the simultaneous matching we construct a graph based on the overlapping areas of the images (see Figure 3). The output of the optimization is a set of new means μ' and standard deviations σ' for each overlapping area that are used afterwards to modify colors accordingly.
In the first step, we determine the overlapping areas in the images. For each image pair, we use the information obtained in the previous visibility computation to identify the areas visible in both views and use them to create two binary images that define the corresponding overlap regions. We create for each overlap region o_{ij} in image I_i that corresponds to an overlap region o_{ji} in image I_j nodes n_{ij} and n_{ji}, respectively. For each node n_{ij} we compute its mean μ_{ij} and standard deviation σ_{ij} using the color values of the associated overlap region o_{ij}. Moreover, we create a color influence map c_{ij} as proposed in [18], that contains for each pixel in the image a weighting factor that describes the influence of the respective node to this pixel. The weight is calculated as the distance of the pixel's color to the color distribution associated with the node. We use the Mahalanobis distance which reduces to

$$ d(p(x), n_{ij}) = \frac{\|p(x) - \mu_{ij}\|}{\sigma_{ij}} $$

since color channels are decorrelated in $l\alpha\beta$ color space. From this distance we compute the entries in the color influence map as $c_{ij}(x) = e^{-3d(p(x),n_{ij})^2}$.
Nodes n_{ij} and n_{ji} of corresponding overlap areas are connected through "outer" edges, i.e. edges between images, whereas "inner edges" e_{ijk} are created between nodes n_{ij} and n_{ik} inside the same image I_i. We use the Weighted-Mean-Variance (WMV) as proposed in [15] to measure the dissimilarity between the color distributions associated with the nodes

$$ cost(e_{ij}) = \left|\frac{\mu_{ij} - \mu_{ji}}{\alpha(\mu)}\right| + \left|\frac{\sigma_{ij} - \sigma_{ji}}{\alpha(\sigma)}\right|, $$

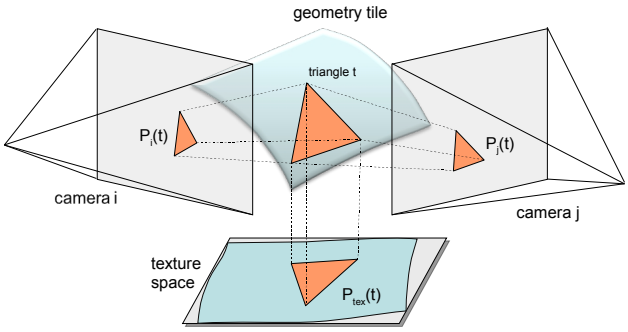

Figure 4. Triangle of a geometry tile and its projections in images and texture domain.

where $\alpha(\mu)$ and $\alpha(\sigma)$ are the standard deviations of the means and standard deviations of all nodes. Edge costs for an outer edge are weighted by the size of the corresponding overlapping areas. The idea behind costs for the outer edges is to penalize differences in the distributions of the color values in the overlapping areas. In contrast to that edge costs for inner edges are weighted based on the inital similarity of the nodes' distributions prior to the optimization. Edge costs for the inner edges aim to avoid a divergence of color distributions associated with different nodes in the same image that were initially similar.
We minimize the sum of all edge costs using the Levenberg-Marquardt algorithm. Using the resulting new means μ' and standard deviations σ', we compute the new color of a pixel $p_i(x)$ in image I_i as a weighted average of the transformations induced by all nodes N_i in the image

$$ p'_i(x) = \frac{1}{|N_i|}\sum_{N_i} c_{ij}(x)\left(\frac{p_i(x) - \mu_{ij}}{\sigma_{ij}}\sigma'_{ij} + \mu'_{ij}\right) + (1 - c_{ij}(x))p_i(x). $$

5.3 Blending

Even after color matching the images do not agree perfectly on a per-pixel level due to small misregistrations or other unmodeled effects. Therefore a good blending strategy is important. A simple approach to blending 2D images is to perform a weighted sum of overlapping color values. However, this approach can cause blurring of high frequency detail if there are small registration errors. To prevent this multi-band blending was proposed in [6]. The basic idea behind multi-band blending is to decompose each image into frequency bands. Each frequency band is combined separately using a weighting function that fits the size of the features in the respective band. The composite bands are finally recombined to obtain the blended image. This

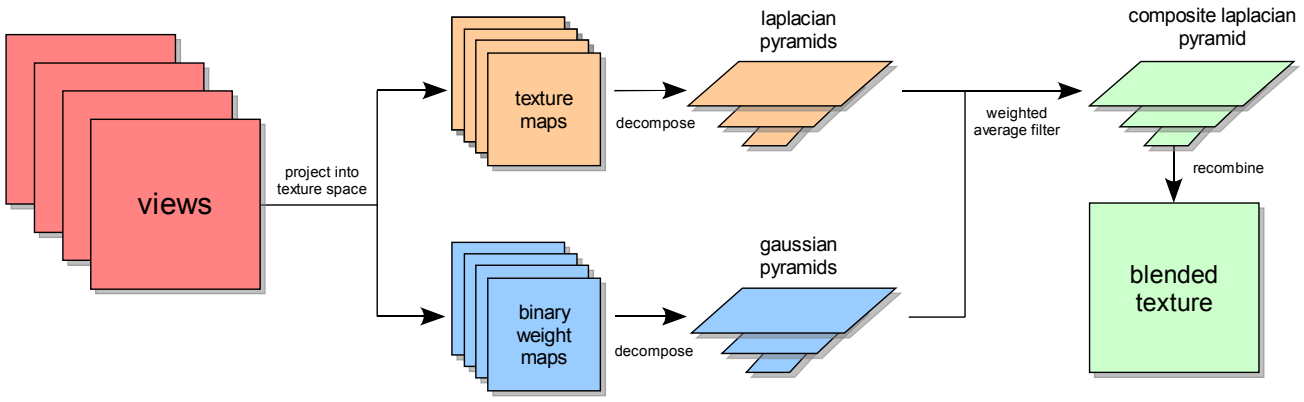

Figure 5. Overview of our patchwise multi-band blending in texture space.

technique allows overlapping images to be blended without introducing visible seams between the images while still preserving the high frequency details and avoiding noticeable ghosting.

In contrast to the aforementioned blending in 2D we need to blend images across a surface in 3D. In order to extend the 2D blending to 3D, we perform a 2D blending in texture space for each geometry tile separately. Given a geometry tile, we identify all cameras it is visible from and create for each a texture and a corresponding weight map in texture space (see Figure 5). Smooth transitions between the photos and the aerial imagery are obtained by including the aerial imagery into the blending procedure as well by considering it as an image from another camera. In order to ensure smooth transitions across tile borders we perform the blending with slightly overlapping patches.

We generate the texture for a certain view by rendering the tile's geometry visible from this view in an offscreen buffer and texture it with the corresponding photo. Holes in the texture caused by occlusion generate artifacts during blending if not handled appropriately. Therefore, we use premultiplied alpha textures where the alpha channel contains visibility information with respect to the considered view. Using premultiplied alpha textures allows us to cancel out the effects of the occluded areas introduced during blending afterwards by dividing the final blended texture by its alpha component.

The corresponding weight maps are created by assigning each pixel a weight $w_{r,i}(x)$ based on the area of the corresponding triangle's projection $P_i(t)$ in the respective image (see Figure 4), reflecting camera position as well as image resolution. In addition to that, we downweight pixels close to an image's border as well as pixels near occluded areas by creating a distance map

$$d_i(x) = \left\| \arg\min_{x'} \{ \|x'\| \mid I_i(x + x') \text{ is not visible } \} \right\|,$$

where each pixel is assigned the distance to the closest pixel not visible in the respective view. We compute corresponding weights from the distances as

$$w_{d,i}(x) = \left(\frac{d_i(x)}{\max_x d_i(x)} \right)^4$$

and multiplicate them pixelwise with $w_{r,i}(x)$. From these weights we create binary weight maps as in [4] by taking the pixelwise maximum.

After the creation of the texture and weight maps we decompose each texture into frequency bands by constructing a laplacian pyramid from it. From each binary weight map we build a gaussian pyramid to form the blending weights for the different bands. A composite laplacian is created from them by a weighted filtering of the different frequency bands. Then we reconstruct the composite laplacian pyramid to obtain the blended texture patch. Finally, we store the resulting texture patch in the corresponding quadtree tile. Optionally, a texture atlas can be created from the texture patch to further reduce memory requirements. However, this requires an extrusion of the triangles' borders to ensure correct texture filtering.

6 Results and Discussion

We have evaluated our method using a HRSC (High Resolution Stereo Camera) data set of Turtmann valley. Turtmann valley is an alpine catchment located in the southern mountain range of the Valais Alps in Switzerland. The data set was acquired in 2001 and contains a 1m DEM

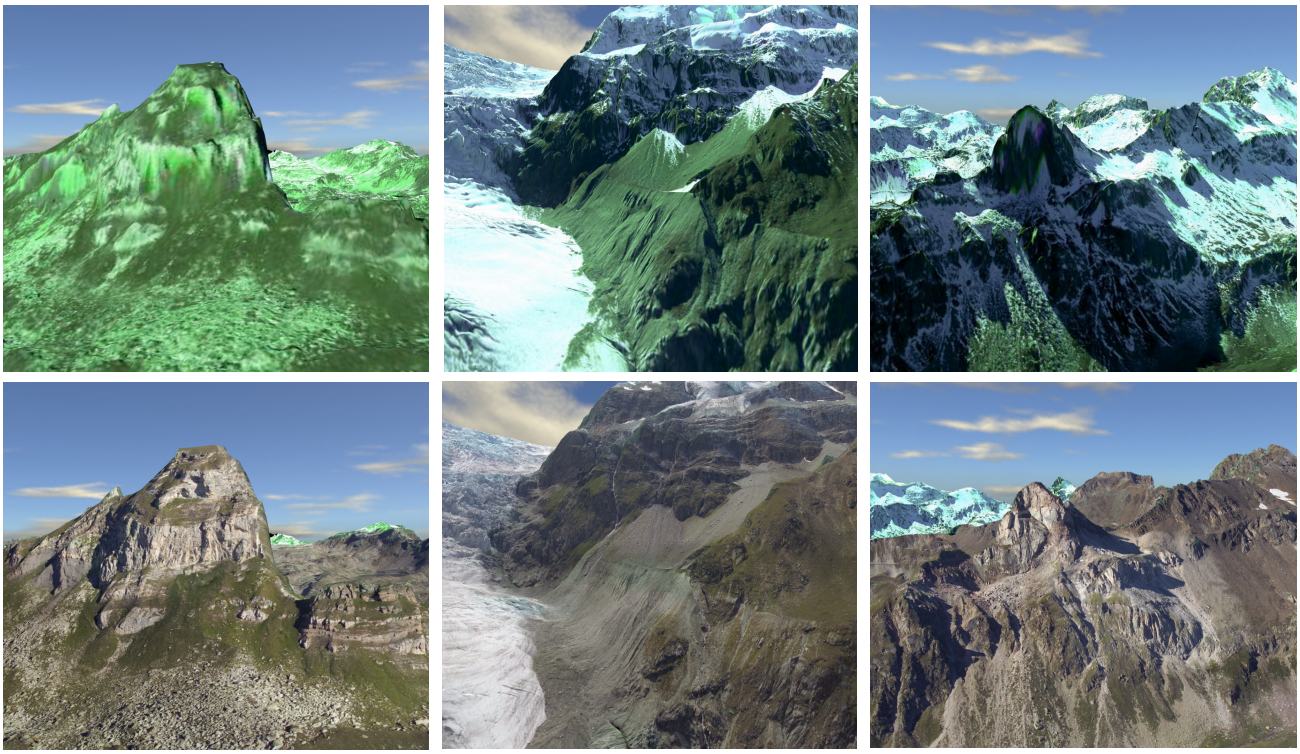

Figure 6. Results obtained with our method. The top row depicts screenshots of the original terrain textured with orthophotos only. The bottom row shows the terrain from the same viewpoints but now textured with the input photos in addition to the aerial imagery.

as well as corresponding aerial imagery at 50 cm resolution covering an area of about 200 km^2 kilometers. The input photo set contains 89 images and was taken by a single person with a single camera in the course of one week. All photos have a resolution of 4500 $\times$ 3000 pixels. Some of the photos were taken from the ground others during a helicopter overflight. Although EXIF tags for the images are available we did not use them in the SfM optimization. After feature matching the whole photo set contained several sets of connected components. Therefore, a manual interaction by the user to provide good initial estimates was necessary for each component. Overall, we were able to match 76 of the 89 input images.

Figure 6 shows an example of the obtained results. The images in the top row depict screenshots of the original ortho-textured DEM where the low texture quality at the steep slopes is clearly visible. In contrast to that, the images in the bottom row show the enhanced DEM obtained from the input photos from the same viewpoint. Compared to the original data set the enhanced representation exhibits a drastically increased visual quality and information content, especially at the steep slopes.

Our method allows a targeted enhancement of already existing terrain data sets. This cannot only be used to increase resolution at steep slopes but also to add information in areas that are not clearly visible in an aerial photo due to shadows, snow coverage or clouds, for example, and such can avoid an expensive re-acquisition of the whole data set. Another interesting application is the acquisition of time varying phenomena. For example, by taking photos of a glacier every year and registering them to the data set it is possible to capture its change over time (see e.g. Turtmann glacier in Figure 6 middle images) without requiring an elaborate geo-referencing of the photos during acquisition.

An issue we did not address in this paper are ghosting artifacts due to occlussion by moving objects or unmodelled terrain geometry. In the future we plan to extent the compositing stage of our approach by a detection and removal of regions of differences [12][28] in the images prior to blending. We also plan to investigate to what extent the geometric accuracy of the DEM can be improved based on the estimated 3D points using multi view stereo

algorithms [10].

7 Conclusions

This paper has presented a method for enhancing a textured DEM using an unordered set of input photographs. Apart from an initial user interaction to provide good estimates to the structure from motion optimization, the input photos were matched to a high-resolution textured DEM automatically. A color matching between the photos followed by a multi-band blending technique created smooth transitions between the photos despite illumination differences while at the same time preserving high frequency details. With the presented method we were able to drastically increase the visual quality and information content of the original data set. Moreover, the insertion of the high resolution textures in the quadtree data structure of a rendering engine allows an efficient real-time visualization.

References

[1] N. Bannai, A. Agathos, and R. B. Fisher. Fusing multiple color images for texturing models. In *3DPVT*, pages 558–565, Washington, DC, USA, 2004. IEEE Computer Society.

[2] A. Baumberg. Blending images for texturing 3d models. page 3D and Video, 2002.

[3] F. Bernardini, I. M. Martin, and H. Rushmeier. High-quality texture reconstruction from multiple scans. *IEEE Transactions on Visualization and Computer Graphics*, 7(4):318–332, October 2001.

[4] M. Brown and D. Lowe. Automatic panoramic image stitching using invariant features. 74(1):59–73, August 2007.

[5] M. Brown and D. G. Lowe. Unsupervised 3d object recognition and reconstruction in unordered datasets. In *3DIM '05: Proceedings of the Fifth International Conference on 3-D Digital Imaging and Modeling*, pages 56–63, Washington, DC, USA, 2005. IEEE Computer Society.

[6] P. J. Burt and E. H. Adelson. A multiresolution spline with application to image mosaics. *ACM Transactions on Graphics*, 2(4):217–236, 1983.

[7] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *Computer Graphics*, 30(Annual Conference Series):11–20, 1996.

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

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, 1981.

[10] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *ICCV 2007*, 2007.

[11] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[12] C. Herley. Automatic occlusion removal from minimum number of images. pages II: 1046–1049, 2005.

[13] H. P. Lensch, W. Heidrich, and H.-P. Seidel. A silhouette-based algorithm for texture registration and stitching. *Graph. Models*, 63(4):245–262, 2001.

[14] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Distinctive image features from scale-invariant keypoints*, 62(2):91–110, 2004.

[15] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. volume 18, pages 837–842, Washington, DC, USA, 1996. IEEE Computer Society.

[16] D. Martinec and T. Pajdla. 3d reconstruction by gluing pairwise euclidean reconstructions, or "how to achieve good reconstruction from bad images". In *3DPVT*, 2006.

[17] D. Martinec and T. Pajdla. Robust rotation and translation estimation in multiview reconstruction. In *CVPR*, 2007.

[18] A. Maslennikova and V. Vezhnevets. Interactive local color transfer between images. In *GraphiCon*, 2007.

[19] W. Niem. Automatic reconstruction of 3d objects using a mobile camera. volume 17, pages 125–134, February 1999.

[20] E. Ofek, E. Shilat, A. Rappoport, and M. Werman. Multiresolution textures from image sequences. *IEEE Comput. Graph. Appl.*, 17(2):18–29, 1997.

[21] F. Pighin, J. Hecker, D. Lischinski, R. Szeliski, and D. H. Salesin. Synthesizing realistic facial expressions from photographs. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 75–84, New York, NY, USA, 1998. ACM.

[22] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, and R. K. K. Cornelis, J. Tops. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

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

[24] C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Multiple textures stitching and blending on 3D objects. In *Eurographics Rendering Workshop*, pages 119–130, 2004.

[25] F. Schaffalitzky and A. Zisserman. Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part I*, pages 414–431, London, UK, 2002. Springer-Verlag.

[26] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Transactions on Graphics*, 2006.

[27] G. Toussaint. Solving geometric problems with the rotating calipers, 1983.

[28] M. Uyttendaele, A. Eden, and R. Szeliski. Eliminating ghosting and exposure artifacts in image mosaics. pages II:509–516, 2001.

[29] R. Wahl, M. Massing, P. Degener, M. Guthe, and R. Klein. Scalable compression of textured terrain data. *Journal of WSCG*, 12(3):521–528, 2004.