The Cartographic Journal *Vol. 48 No. 1 pp. 11–20 February 2011*

REFEREED PAPER

Interactive Local Terrain Deformation Inspired by Hand-painted Panoramas

Helen Jenny[1], Bernhard Jenny[1], William E. Cartwright[2] and Lorenz Hurni[1]

[1] Institute of Cartography, ETH Zurich, Switzerland. [2]Geospatial Science, RMIT University, Melbourne, Vic., Australia

Email: helen.jenny@karto.baug.ethz.ch

Painters of panoramic landscape maps use specific manual techniques to solve problems of occlusion, foreshortening and unfavourable orientation of landscape elements to the map viewer. Using digital means, the painters' techniques may be translated into geometry deformation algorithms for digital panorama creation. This article explores the advantages and the suitability of applying local geometry deformation to digital panoramas and reviews existing methods to perform such terrain editing with digital means. A new algorithmic solution based on inverse distance interpolation and moving least squares and specifically designed for regular 2.5D elevation models is presented. It allows the user to position and drag control handles on a 3D representation of the model to interactively deform the terrain.

Keywords: terrain distortion, panorama, H. C. Berann, 3D map design, inverse distance, moving least squares

INTRODUCTION

Static panoramic maps commonly show the landscape from a specific point of view and usually in central perspective projection. No matter how thoughtfully the point of view is chosen, some important landscape elements will be foreshortened, occluded or shown from an unfavourable angle to the observer. Landscape artists have encountered such challenges by applying local deformation to map objects in hand-painted hiking and skiing panoramas. In digitally created panoramic maps, local terrain deformation is largely absent, presumably because the functionality of standard 3D rendering software does not sufficiently support the needs of cartographers.

In the following article, we draw inspiration for digital cartographic terrain deformation from the works of panorama painter H. C. Berann (1915–1999) (Patterson, 2000). We explore the advantages of applying local deformation to digital panoramas and review existing methods to perform such terrain editing with digital means. A new algorithmic solution based on inverse distance interpolation and moving least squares is subsequently presented. The authors extended Terrain Bender (Jenny *et al.*, 2010), a software for global terrain deformation, to include this new local method. The cartographer can locally deform a digital terrain model by intuitively manipulating the surface in a 3D display. We created a series of digital panoramic maps, with and without deformation, showing a region for which a hand-painted panorama is available. Such a comparative series demonstrates well the effectiveness of our method and illustrates the terrain deformations applied by H. C. Berann in his work.

Painters of panoramic maps often use local deformation and global progressive bending of the terrain as complementary manual techniques. Figure 1 illustrates this combined approach described by contemporary panorama artist Juan Nuñez Guirado (Ribas Vilas and Nuñez Guirado, 1990). Progressive terrain bending (Figure 1d) allows for landscape displays with varying viewing angle from steep in the foreground to flat in the background. As we have already provided a digital method to create panoramas with progressive bending (Jenny *et al.*, 2010), the method for local terrain deformation (Figure 1c) presented here completes bringing the manual geometry deformations to the digital realm.

REPRESENTATION CHALLENGES OVERCOME BY APPLYING LOCAL DEFORMATION

Experienced panorama painters apply local deformation to solve specific representation problems. The following overview of use cases is based on Patterson's (2000) analysis of the works of H. C. Berann. Three problem groups can be recognized resulting from occlusion, foreshortening and orientation to the viewer.

Occlusion: Panoramas show the landscape from an oblique viewing-angle. As a consequence, map objects depicted in the foreground can partially or totally hide those located behind them. This can obviously be a problem on touristic maps, the major application field of panoramic maps: hiking

DOI: 10.1179/1743277411Y.0000000002

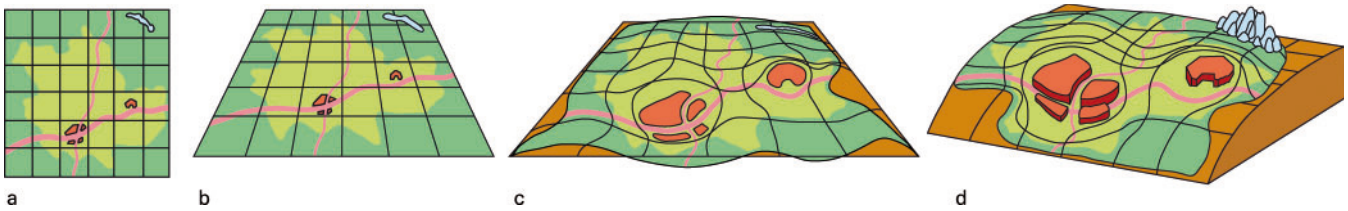

Figure 1. Combination of local deformation and progressive bending in manual panorama creation: landscape in orthogonal view (a), in central perspective projection (b), with local deformations (c) and with local deformations and progressive bending (d) (Ribas Vilas and Nuñez Guirado, 1990, simplified, colours adapted)

and skiing trails may be hidden behind mountains or forest; important landmarks and touristic infrastructure are concealed behind river and valley bends. Panorama painters mitigate these occlusions by locally deforming the landscape, e.g. by moving occluding mountains, widening valley floors, straightening river bends, enlarging landmarks and levelling terrain.

Foreshortening: Most panoramas are drawn in central perspective projection to mimic the way human eyes perceive the world. As a consequence, foreshortening in the background can reduce landscape parts beyond recognition. Touristic infrastructure, reference landmarks and trails may also not be discernable. In addition, the landscape panorama may lack in aesthetics and focus when well-known mountain ranges are shrunk. To accentuate important landscape features and to compensate for foreshortening, panorama painters enlarge selected landscape elements that would otherwise receive too little attention from the observer.

Orientation to the viewer: Selecting the observer position for a panorama is a challenging task. Even if it is well chosen, not all important landscape parts and landmarks will be shown from their most familiar or impressive side. Some mountain silhouettes have been used widely in touristic or commercial contexts (e.g. the Swiss Matterhorn) and may be recognized by the public only from this often-depicted side. Panorama painters rotate reference landscape elements to depict them from a better-known or more informative side.

Modern mapmakers who use digital tools to create panoramic maps also encounter the described representation problems. We think that digital algorithms, which draw from the solutions of the panorama painters, would allow cartographers to create better digital panoramas.

REQUIREMENTS ON A LOCAL DEFORMATION METHOD FOR DIGITAL PANORAMAS

Using digital means, the techniques applied by the panorama painters may be translated into geometric transformations of the digital terrain surface. In the following section, we discuss what makes a good method for local terrain deformation and suggest a number of requirements.

According to Botsch and Sorkine (2008), the quality of a surface deformation method depends on its intuitive handling, its flexibility and the quality of the shapes it produces. They suggest that an intuitive deformation behaves in some ways like a real world object underlying physical principles. Yet, for design or visualisation purposes as opposed to simulation, the results only need to be plausible, aesthetic and correspond to what the user would naturally expect. It is not necessary to consider every physical constraint or to permit arbitrary surface editing as this may make results difficult to control for the user. For the cartographer, it is also essential to inspect and adjust different deformation variants quickly and easily, since the evaluation of local deformation is mainly carried out visually.

Based on these requirements, we suggest that a local deformation method for digital panoramas is sufficiently intuitive if the user can manipulate the terrain in a 3D view; if deformations are computed and displayed immediately; if interactive deformation appears to work directly on the terrain surface; and if point-to-point correspondences between the original and the deformed model can be defined. Concerning flexibility, our method should allow for shifting landscape elements in the horizontal plane, and for rotating and scaling landscape elements. The shape of the landscape after deformation should still appear natural; characteristic details should be preserved; and transition zones between deformed and undistorted areas should look smooth and inconspicuous.

Digital elevation models used in cartography often consist in millions of altitude values. Our method needs to be fast enough to compute deformation previews of such terrain data in real-time on desktop computers. It should also be possible to export deformed terrain models to further process them with specialized software. Adapting the coordinates of an accompanying texture for export is also required, since otherwise, the original texture would not fit the deformed model anymore. Before describing our new method to design local terrain deformations, however, the following section reviews related work.

RELATED WORK

3D terrain cartography shares interests with geographic information science and computer graphics. The author of a panorama map requires methods that support handling georeferenced, potentially large digital elevation models of real landscapes while providing at the same time interactive freedom of design. We evaluate the potential of standard GIS and 3D rendering software before giving an overview of terrain deformation implemented in research systems.

Standard GIS and 3D rendering software

In standard GIS, terrain model editing concentrates on enforcing real geometry (e.g. roads and spot elevations) in

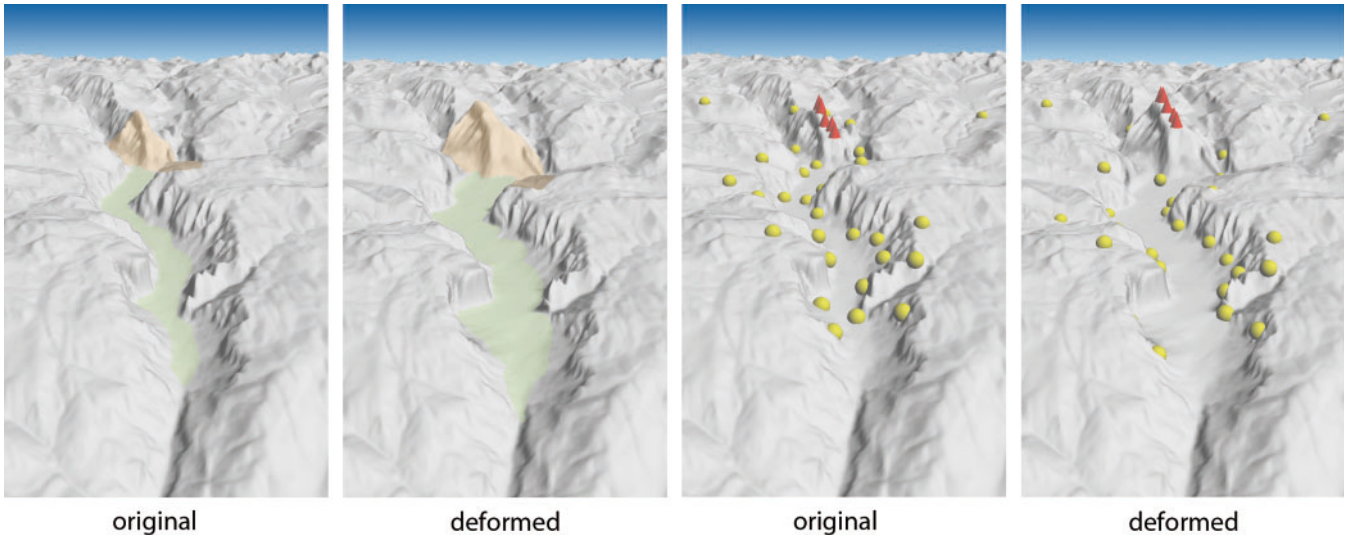

Figure 2. Examples for local deformations achieved with our method in the Yosemite Valley. Left pair: vertical scaling of Half Dome (orange), widening of a valley (green) and levelling of terrain (brown); right pair: control point location before and after deformation (based on USGS DEM)

georeferenced terrain in a topologically correct way and on interpolating terrain models from surveying data. Since the user usually provides such real-world information as georeferenced datasets, interactive editing is limited and provides little freedom of design.

On the other hand, standard 3D modelling and rendering software offers a powerful range of interactive deformation and editing options for 3D meshes, many of them specialized (e.g. methods for character rigging and facial expression manipulation). Yet, these methods are not targeted at georeferenced objects and often cannot handle terrains with millions of elevation values. Also, the learning curve is often steep, as experience is needed to predict the provided tools' effect (e.g. paintbrush to edit elevations) and to switch between view and parameter windows. Software for 3D game creation usually offers diversified options to create artificial terrain, but provides only limited support for editing real terrain as procedural terrain is often preferred for real-time games.

Terrain editing

Atlan and Garland (2006) describe a multi-resolution editing system for terrains with real-time display based on wavelets. Editing tools use metaphors of landscape architecture, e.g. bulldozer and explosion tools. Bruneton and Neyret (2008) propose a method for blending vector data into height maps using shader programs running on graphics hardware. The height field is edited indirectly by adapting the vector data. Clark and Mauer (2006) propose reinterpolating terrain by integrating survey points and interactively moving height field points in vertical direction.

Digital panoramas

Möser *et al.* (2008) propose multi-perspective and importance scaling techniques to increase visibility in 3D urban panoramas. Falk *et al.* (2007) implemented a projective ray curving approach using a force field. Local occlusions in panoramic maps can be reduced by letting the user paint influence textures to modify the force field. Takahashi *et al.* (2006) reduce occlusions occurring along the route in a car navigation system by vertical scaling of the terrain. Degener and Klein (2009) automatically deform a terrain surface using a variational formulation to maximize the visibility of a specific set of features.

A METHOD FOR LOCAL DEFORMATION OF 2.5D TERRAIN SURFACES

Digital elevation models used in cartography are often regular 2.5D models, also called functional surfaces. For every grid cell, only one elevation value is stored. Some 3D shapes requiring more than one altitude value per cell (e.g. overhanging cliffs) thus cannot be represented in 2.5D. Yet, regular functional surfaces are widely used in cartography and GIS since they are easy to process and to combine with other gridded content like photogrammetric imagery or thematic grids.

Our method for local terrain deformation is especially designed for such regular 2.5D elevation models. The user positions handles in the form of control points on the model, which can be dragged to deform the terrain. Our deformation functions provide a mapping for every grid point v in the original terrain model to a position in the deformed model, based on a set of control points p and their new positions q. Grid points coincident with control points p are coincident with positions q after deformation. The remaining terrain is deformed smoothly by computing a displacement vector for every grid point based on the displacement of the control points. Figure 2 shows example deformations achieved with our method by dragging control points. Our deformation method allows varying the radius of influence of control points between more local and more global influence.

We implemented two computational methods to locally deform a digital terrain model: inverse distance weighting

(IDW) in three dimensions and, as an alternative, moving least squares minimisation (MLS).

Inverse distance interpolation

IDW as defined by Shepard (1968) is a multivariate interpolation method for scattered points. It assumes that the influence of the N control points to induce terrain deformation decreases with the distance between control points and grid points. The scaled position of an elevation point v is computed by moving it by the displacement vector $\overrightarrow{d_v}$, which is determined using equation (1)

$$\overrightarrow{d}_v = \sum_{i=1}^{N} \frac{r_i(v)}{\sum_{j=1}^{N} r_j(v)} \left(\overrightarrow{q_i} - \overrightarrow{p_i} \right)$$

$$\text{where } r_i(v) = \frac{1}{\left| \overrightarrow{v} - \overrightarrow{p_i} \right|^t} \text{ with } t \geq 1 \qquad (1)$$

The weight r in our algorithm is the inverse distance in the undistorted model between the grid point v and the control point p_i. A faster rate of distance decay may be achieved by increasing t, giving more influence to the nearest control points and increasingly down-weighing points farther away. The grid points at control point position p_i take on the location q_i after deformation.

MLS minimisation

In addition to the IDW method described in the previous section, we implemented another method for manipulations in the horizontal plane. It builds on an algorithm by Schaefer *et al.* (2006) who use MLS to apply affine transformations for 2D image deformation. We use Schaefer's rigid transformation for terrain deformation in the plane and combine it with IDW for vertical deformation.

According to Schaefer *et al.* (2006), the plane is distorted by solving for the optimal affine transformation f_v that is applied to all grid points v and minimizes

$$\sum_i w_i |f_v(p_i) - q_i|^2 \qquad (2)$$

This means that the weighted sum of squared distances between the desired control point positions q and their computed position using $f_v(p)$ is as small as possible.

The weights w_i depend on the distance of the grid points to the control points, with small distances being of larger significance. As a consequence, the transformation is different at each grid point. The parameter α describes how strongly the influence of the control points diminishes with increasing distance:

$$w_i = \frac{1}{|p_i - v|^{2\alpha}} \qquad (3)$$

Schaefer *et al.* (2006) propose three transformations: the affine, similarity and the rigid deformation. As they suggest that rigid deformation gives the best results for realistic shapes, we have used this warping function for our method. Schaefer *et al.* (2006) find the deformation function f_v (equation (4)) for every grid point v, involving only small linear systems with closed form solutions.

The weighted centroids p^* and q^* of the sets of control points p and q, as well as the deformation function f_v are computed with:

$$p^* = \frac{\sum_i w_i p_i}{\sum_i w_i} \text{ and } q^* = \frac{\sum_i w_i q_i}{\sum_i w_i}$$

$$f_v = (v - p^*)M + q^* \qquad (4)$$

For rigid deformation, M is

$$M = \frac{1}{\mu} \sum_i w_i \begin{pmatrix} \hat{p}_i \\ -\hat{p}_i^{\perp} \end{pmatrix} \left(\hat{q}_i^{T} - \hat{q}_i^{\perp T} \right) \qquad (5)$$

where $\perp$ is an operator that maps a vector (x,y) to $(-y,x)$, $\hat{p}_i = p_i - p^*$ and $\hat{q}_i = q_i - q^*$, and

$$\mu = \left[\left(\sum_i w_i \hat{q}_i \hat{p}_i^{T} \right)^2 + \left(\sum_i w_i \hat{q}_i \hat{p}_i^{\perp T} \right)^2 \right]^{1/2} \qquad (6)$$

INTEGRATION INTO TERRAIN BENDER

Panorama painters apply local deformation of terrain often in combination with progressive bending. For this reason, we chose to extend the functionality of Terrain Bender, a software framework previously developed to apply progressive bending to digital elevation models (Jenny *et al.*, 2010) with local deformation algorithms. Terrain Bender was written in Java and runs on Windows and Mac OS X platforms. For rendering previews of digital elevation models, Terrain Bender uses JOGL, a Java wrapper library for OpenGL (JOGL, 2009). OpenGL is a standard specification, defining an application programming interface for generating 2D and 3D computer graphics (OpenGL, 2010).

Local terrain manipulation

The cartographer can import a 2.5D digital terrain model into Terrain Bender, which is then displayed in a 3D preview. By clicking with the mouse on the rendered terrain surface, the cartographer can set control points and can apply local deformation by dragging the points to new positions. The number of elevations in the model does not change and intermediate deformation poses are not stored. Instead, to undo a deformation step, a control point can be deleted or deformation vectors can be reset, triggering a re-computation of the deformed grid. It is possible to add control points gradually to an already deformed grid. The resulting distorted grid is still topologically regular, but has irregular geometry.

In our prototype, four types of control handles are available. Regular control points are symbolized as spheres (Figure 2, right image pair) and can be dragged in all three directions. If not moved, they act as counterweights to

Figure 3. Jungfraubahn (H. C. Berann, 1947). Hand-painted panorama showing the Jungfrau region in Switzerland

other dragged control points. Vertical handles represented by cones are restricted to movements in the vertical direction (Figure 2, red cones in right image pair). They do not act as counterweights to modifications in the horizontal plane and are ignored when horizontal displacements are computed. Circular belt symbols represent attracting or repelling handles that radially pull or push the surrounding grid points towards or away from them. One or several control points at a time can be selected and dragged as a group to deform the terrain.

Large terrain models and graphics processing unit acceleration

Standard 3D modelling and rendering software often cannot handle large, memory intensive terrain models. Terrain Bender builds a Gaussian pyramid from the original model made of successively smaller grids. Each grid cell contains a local average of cells from a more detailed level (Jenny *et al.*, 2010). The user can choose a lower resolved pyramid level to speed up the previewing response. Local deformations are then applied to the downsampled grid. Since we store the displacement of control points to compute the deformation of the entire grid, it is straightforward to compute a deformation for a different resolution. To further improve performance, we use a graphics processing unit implementation where a vertex shader computes the deformed positions of the model to be displayed. The vertex shader is a processing function executed on graphics hardware, which is run for each elevation grid point. The original grid point positions need to be transferred only once to the graphics card and can then be modified using the vertex shader. This is a fast solution for rendering when only positions need to be modified without changing the number of vertices as in the case of our local deformation algorithm.

Model export and textures

After deformation, the elevation model has still the same number of grid quads, but the geometry of these quads is not regular anymore. In Terrain Bender, the user has the option to drape a texture on the elevation model. The deformed model can then be exported for further processing and rendering in triangulated form together with texture coordinates.

IMITATION OF A HAND-PAINTED PANORAMA

We demonstrate our local deformation method by imitating a manual panorama by H. C. Berann (Figure 3). Berann's panorama was painted in 1947 and shows the Jungfrau region with hiking trails, cable cars and other touristic infrastructure. Figures 4–6 show the digital panorama at different deformation stages and permit comparisons between the painted panorama, our deformed digital result

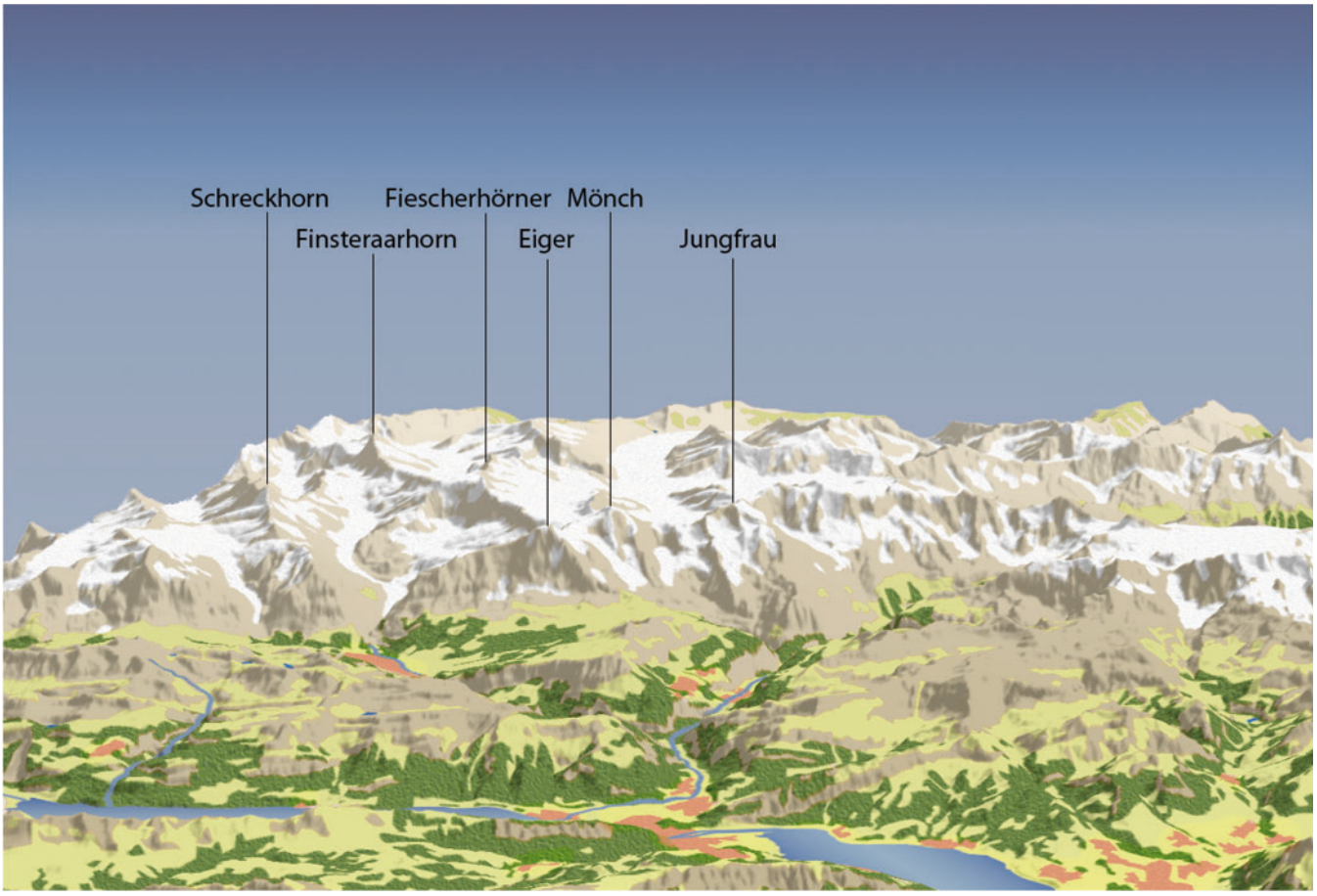

Figure 4. Undeformed digital rendering with central perspective projection of the Jungfrau region depicted in Figure 3

and the undeformed digital model. Please note that we only intend to imitate the geometry deformation; panorama texture generation is not in the scope of this article and we only apply standard texturing techniques to add a simple land cover texture.

Figure 4 shows a digital rendering of approximately the same region as in the hand-painted panorama in central perspective projection without any alteration to the elevation model. In Figure 5, we applied progressive bending and little vertical background scaling with Terrain Bender. We used a swisstopo digital elevation model with 25 m resolution and a land use texture. The foreground lakes are presented from a steep viewing angle and are less occluded compared to the undeformed rendering in Figure 4. In the background, the mountain ranges, which are depicted from a flat viewing angle, now form a horizon. Yet, this mountainous horizon appears unstructured: many peaks have similar elevation and higher peaks have small width, which does not give them enough weight to dominate the landscape. In Figure 6, we combined progressive bending and vertical background scaling with local distortion. Eiger, Mönch and Junfrau mountains were scaled vertically and horizontally to give the panorama a visual focus. The Fiescherhörner and the Finsteraarhorn to the left of Eiger (still visible in Figure 5) were rotated clockwise to bring them behind Eiger so that they would not appear on the panorama. A small folding at the mountain base where Eiger meets the Schreckhorn remains. The observer can now see the valley floors in Figure 6, which are mostly occluded in Figures 4 and 5. This was achieved by either broadening the valleys or by lowering the terrain in front of them. The lakes were also rotated so that they would be more parallel to the lower panorama border imitating the lakes' positions in Figure 3.

ROTATION OF A MOUNTAIN RANGE USING IDW AND MLS MINIMISATION

The Greater Yellowstone panorama (Figure 7) by H. C. Berann is a convenient test case to compare the results of rotating a mountain range with the IDW and MLS methods. As Patterson (2000) pointed out, Berann rotated the Teton Range in the background of the Greater Yellowstone Panorama by about 55°.

In Figure 8, we show the Teton Range in the Yellowstone Region in their original shape and location and rotated with IDW and MLS. We imitate only the background of the painted panorama concentrating on the Tetons (Figure 7, red box). The thick arrow on the orthogonal representation of the region (Figure 8, top left) shows the approximate viewing direction of the panorama.

In the digital panorama without rotation (top right), the Tetons are only visible as a stub in the background and a flat

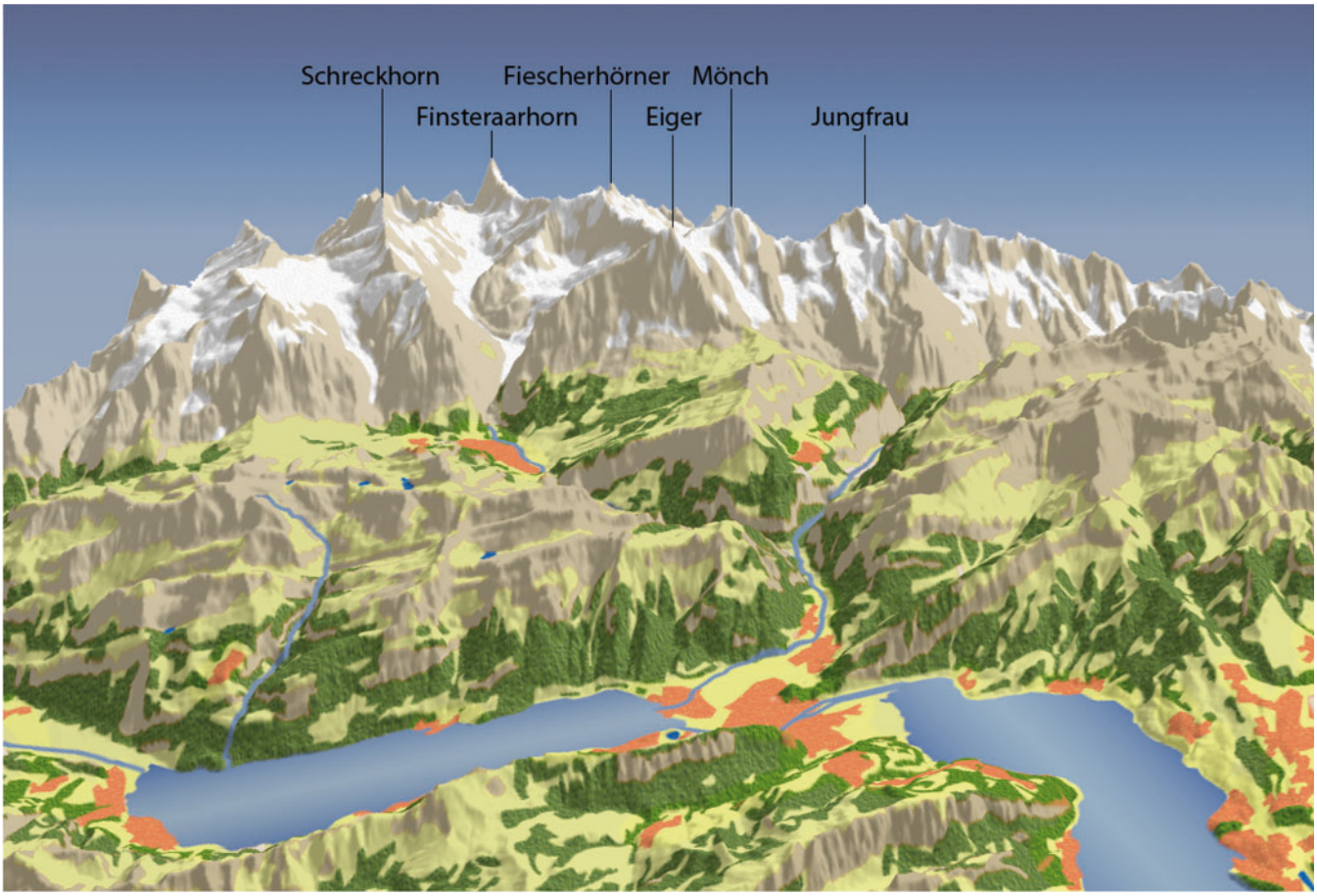

Figure 5. Digital panorama of the Jungfrau Region with central perspective projection, vertical exaggeration in the background, and progressive bending applied

area forms the centre of the background. In the middle and bottom representations, we rotated the Tetons to imitate Berann's representation using six control points shown in red. Only the topmost control point was dragged in the horizontal plane to apply deformation, while the other control points acted as counterweights. To visualize the deformation, we added vector fields to the orthogonal representation of the test area (middle left and bottom left). They confirm that the deformation produced with MLS resembles more closely a rotation than the deformation computed with IDW. The characteristic shape of the Tetons is better preserved with MLS compared to IDW, where self-intersecting geometry is created by the deformation. In the digital IDW panorama, the degenerate geometry is visible as grey artefacts in the left background.

Often, we have found that for less extreme deformations, oblique views generated with IDW and MLS are of equal visual quality. In the previous example, if the rotation angle is reduced only by a few degrees, no self-intersections are created with IDW and one cannot visually tell the difference between the IDW and the MLS panoramas. Yet, the orthogonal views still show obvious differences, with the MLS representation performing better. This may be because detailed differences cannot be well discerned in the background of the panoramas, but the general shape is preserved with both methods. With very large rotation angles, the MLS method also creates geometric overlaps.

DISCUSSION AND FUTURE WORK

A shortcoming of our method is the possibility to create geometrically degenerate models, corresponding to self-intersections of the terrain caused by extreme deformation. Such degenerate geometry is most often produced when rotating landscape elements or strongly compressing the panorama background. Larger areas of degenerate geometry are usually easy to spot. Schaefer *et al.* (2006) suggest that fold-backs caused by their 2D deformation function could be mitigated by applying an approach by Tiddeman *et al.* (2001) which was implemented for metro map warping by Böttger *et al.* (2008). Such measures to avoid creating degenerate geometry may also be helpful for our method.

Another problem is inherent to weighted distance-based approaches. When Euclidean distances are used for deformation, the shape of the object is not considered. As an example, if we imagine mountains in the shape of two very close tall spikes, manipulating one mountaintop has a stronger influence on the other mountain's top than on its base. Yet, the user would expect the influence to be greater on the base because the distance to the base within the hull of the terrain is smaller than the distance to the next top. Cuno Parari *et al.* (2009) use an approach based on a skeleton, and Zhu and Gortler (2007) use a vertex visibility-based approach to measure inner distance in moving least squares methods. Our algorithm could benefit from such an inner distance measure for elevation models.

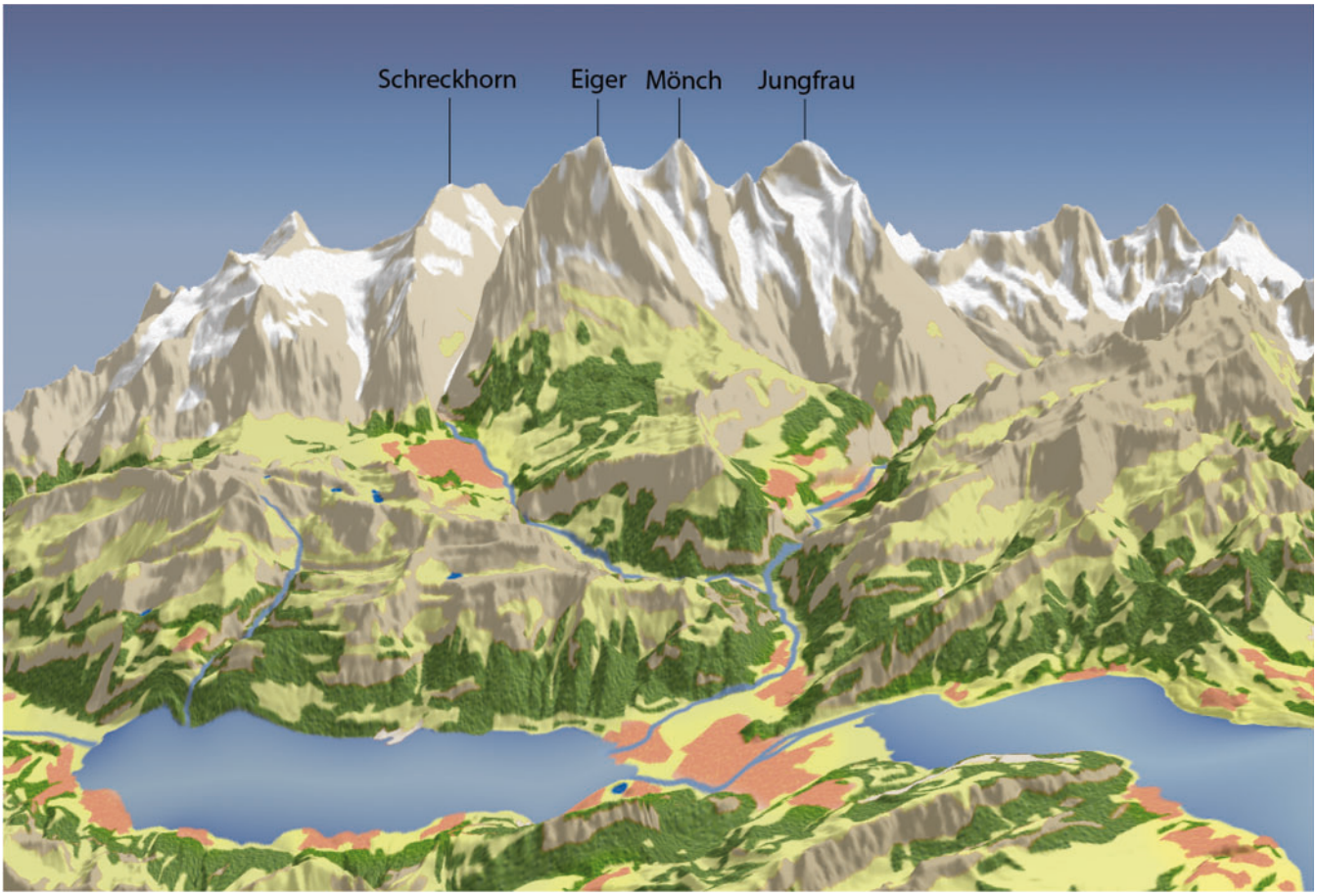

Figure 6. Digital panorama with additional local deformation applied, IDW method with 71 control points

Various approaches exist for improving Shepard's inverse distance method. For example, Franke and Nielson (1980) improve the interpolation results by reducing the impact of control points on far away samples. For our application of terrain deformation, it would be interesting to find out if setting a maximum influence radius of the control points would improve the surface shape. We found that our global method handles transition zones well; yet, a global algorithm has the disadvantage of shifting all points when a control point is added or moved, even if this influence is not desired.

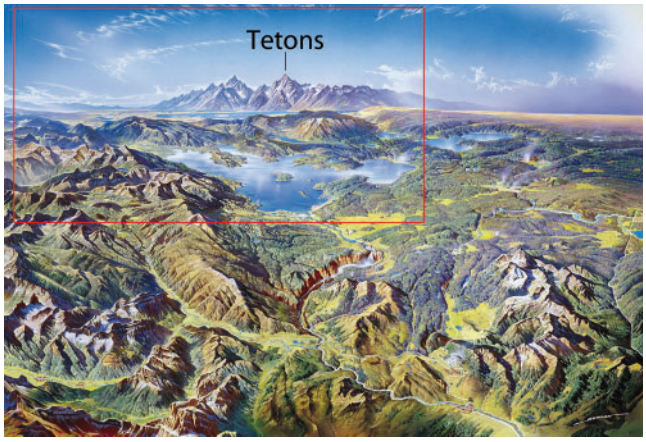

Figure 7. Greater Yellowstone. Hand-painted National Park Panorama by H. C. Berann, 1991. The region to be imitated depicting the Tetons is marked with a red box

Our method is also not aware of landscape units and their characteristics. A deformed lakeside may therefore locally shift in vertical direction and appear to wander up a hill, or a deformed river may appear to flow upwards. It would be interesting to derive additional constraints from land use datasets to automatically avoid such disturbing effects. More user-friendliness and better results could be achieved by including control lines in the method in addition to control points.

We also found that it is much easier to imitate hand-painted panoramas where the observer looks up towards the mountains with only one or two ranges dominating the background view (e.g. Figure 6). Panoramas where the observer looks towards the horizon from an airplane-like perspective often show many consecutive mountain ranges in the background. Since in a central perspective view these ranges are displayed on comparatively small image space, it is difficult to edit the control points and often requires switching between different inclination angles and viewing distances. The possibility to view the terrain model simultaneously in additional windows with different viewing parameters would facilitate this task. Also, compressing the background ranges as strongly as the painters did was often impossible without creating degenerate terrain geometry. A multi-scale grid or terrain generalisation approach where the background has a lower resolution than the foreground would be helpful to reduce disturbing geometric detail if necessary.

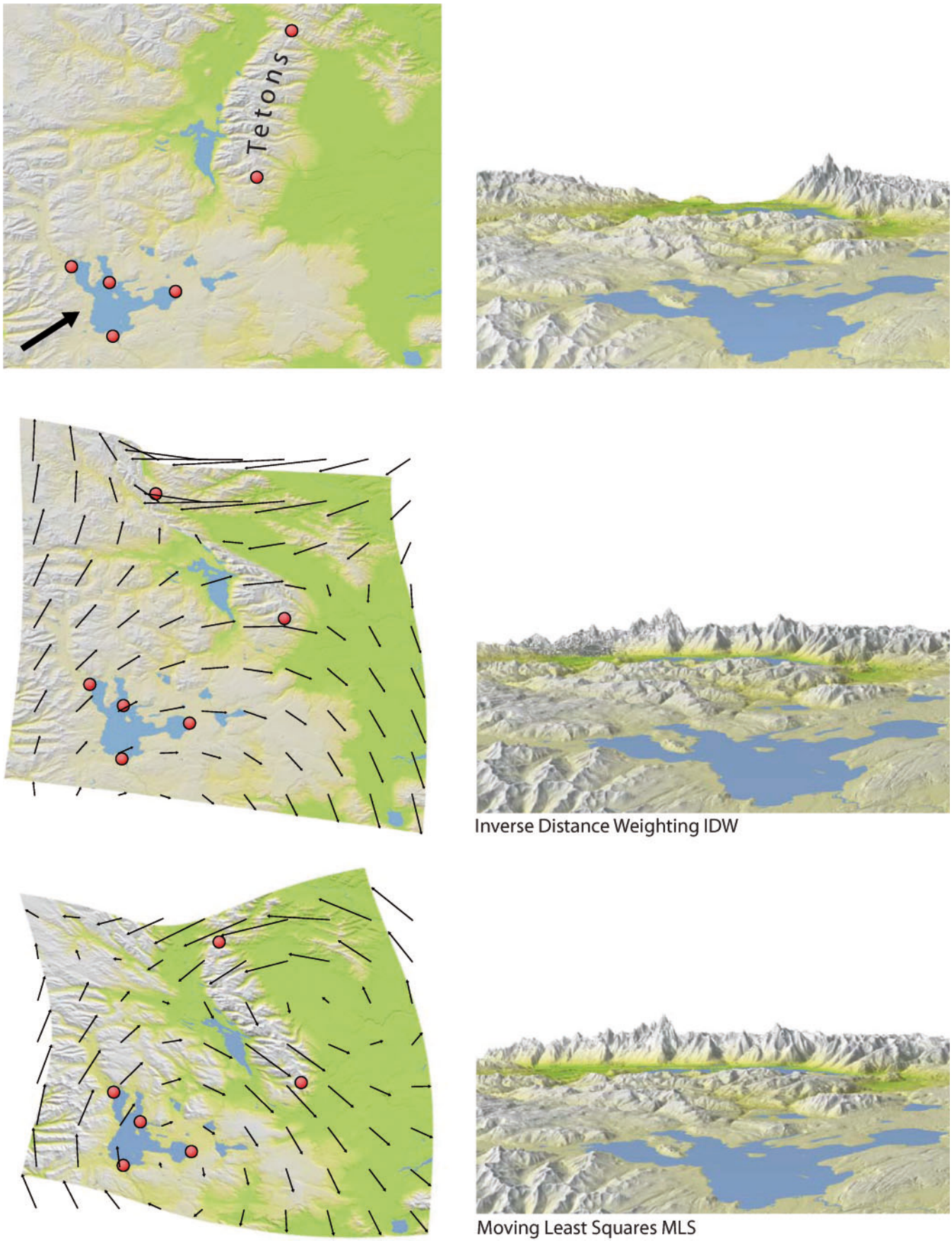

Figure 8. Rotation of the Tetons: original region (top), deformed region using IDW (middle), deformed region using MLS (bottom); orthogonal views (left column), panoramic views (right column). Control points red

CONCLUSION

Our method for local terrain deformation allows the cartographer to solve typical problems occurring when creating panoramic maps. The algorithm to mitigate problems related to occlusion, foreshortening and orientation of landscape elements was inspired by hand-painted panoramas. Yet, with our algorithm, the cartographer does not need the artistic talent of the panorama painters. Terrain deformations can be intuitively achieved by placing and dragging control points on a 3D terrain display. Results are immediately generated for visual evaluation by the mapmaker. In contrast to standard 3D rendering and modelling software, our software for local terrain deformation is targeted specifically at large 2.5D terrain surfaces. Deformed terrain models with textures can be exported for further processing and rendering. The manipulation of the graphical user interface, based on control points placed directly onto the model, is also user-friendly and quick to master.

A better digital imitation of the techniques of panorama painters would be achieved if in addition to our geometry deformation algorithms, specialized panorama textures and terrain generalisation dependent on the viewing distance could be included. It would be interesting to combine our local terrain deformation and progressive bending algorithms with painterly rendering methods for panoramas by Bratkova *et al.* (2009) and terrain geometry generalisation methods. With such a combination, it would be possible to evaluate how digital panorama creation methods, including generalized terrain and texture, compare to hand-painted panorama techniques.

BIOGRAPHICAL NOTES

Helen M. Jenny is a PhD student and a member of the research staff at the Institute of Cartography, ETH Zurich, Switzerland. She received a MS in Physical Geography and GIS from the University of Stuttgart, Germany. Her current research interests are 3D map design, distortions and painterly rendering of digital landscape panoramas, web mapping and map portals.

ACKNOWLEDGEMENTS

We thank Juan Nuñez Guirado, panorama artist, and Rafael Roset, Cartoteca Digital, Institut Cartogràfic de Catalunya, Barcelona, Spain, for providing the basis for Figure 1; Mohammad Hammoudeh, Manchester Metropolitan University, for providing code and information on the Shepard interpolation; Matthias Troyer, ETH Zurich, for providing Figure 3; and Tom Patterson, US National Park Service, for providing Figure 7.

REFERENCES

Atlan, S. and Garland, M. (2006). 'Interactive multiresolution editing and display of large terrains', **Computer Graphics Forum**, 25, pp. 211–224.

Botsch, M. and Sorkine, O. (2008). 'On linear variational surface deformation methods', **IEEE Transactions on Visualization and Computer Graphics**, 14, pp. 213–230.

Böttger, J., Brandes, U., Deussen, O. and Ziezold, H. (2008). 'Map warping for the annotation of metro maps', **IEEE Computer Graphics and Applications**, 28, pp. 56–65.

Bratkova, M., Shirley, P. and Thompson, W. B. (2009).'Artistic rendering of mountainous terrain', **ACM Transactions on Graphics**, 28, pp. 1–17.

Bruneton, E. and Neyret, F. (2008).'Real-time rendering and editing of vector-based terrains', **Computer Graphics Forum**, 27(2), pp. 311–320.

Clark, R. W. and Mauer, R. (2006). 'Visual terrain editor: an interactive editor for real terrains', **Journal of Computing Sciences in Colleges**, 22, pp. 12–19.

Cuno Parari, A. E., Esperança, C. and Oliveira, A. (2009). 'Shape-sensitive MLS deformation', **The Visual Computer**, 25, pp. 911–922.

Degener, P., and Klein, R. (2009). 'A variational approach for automatic generation of panoramic maps', **ACM Transactions on Graphics**, 28(1), pp. 1–14.

Falk, M., Schafhitzel, T., Weiskopf, D. and Ertl, T. (2007). 'Panorama maps with non-linear ray tracing', in **Proceedings of the 5th International Conference on Computer Graphics and Interactive Techniques in Australia and Southeast Asia (GRAPHITE'07)**, ACM, New York, pp. 9–16.

Franke, R. and Nielson, G. (1980). 'Interpolation of large sets of scattered data', **International Journal for Numerical Methods in Engineering**, 15, pp. 1691–1704.

Jenny, H., Jenny, B. and Hurni, L. (2010). 'Interactive design of 3D maps with progressive projection', **The Cartographic Journal**, 47, pp. 211–221.

JOGL. (2009). Java™ Binding for the OpenGL® API Wiki, http://kenai.com/projects/jogl/pages/Home (accessed 28 October 2010).

Möser S., Degener, P., Wahl, R. and Klein, R. (2008). 'Context aware terrain visualization for wayfinding and navigation', **Computer Graphics Forum**, 27, pp. 1853–1860.

OpenGL.org. (2010). OpenGL – The Industry's Foundation for High Performance Graphics, http://www.opengl.org (accessed 28 October 2010).

Patterson, T. (2000). 'A view from on high: Heinrich Berann's panoramas and landscape visualization techniques for the US National Park Service', **Cartographic Perspectives**, 36, pp. 38–65.

Ribas Vilas, J. and Nuñez Guirado, J. (1990). 'Cartografia perceptiva del paisaje: el plano grafico', **Monografies de l'EQUIP**, 3, pp. 293–297.

Schaefer S., McPhail, T. and Warren, J. (2006). 'Image deformation using moving least squares', **ACM Transactions on Graphics**, 25, pp. 533–540.

Shepard, D. (1968). 'A two-dimensional interpolation function for irregularly-spaced data', in **23rd ACM National Conference**, pp. 517–524, New York, Aug 27–29.

Takahashi S., Yoshida, K., Shimada, K. and Nishita, T. (2006). 'Occlusion-free animation of driving routes for car navigation systems', **IEEE Transactions on Visualization and Computer Graphics**, 12, pp. 1141–1148.

Tiddemann, B., Duffy, N. and Rabey, G. (2001). 'A general method for overlap control in image warping', **Computers and Graphics**, 25, pp. 59–66.

Zhu, Y. and Gortler, S. (2007). '3D deformation using moving least squares', Harvard Computer Science Technical Report, Harvard University, Cambridge, MA, USA.