High-Quality Cartographic Roads on High-Resolution DEMs

Mikael Vaaraniemi
BMW Forschung und Technik GmbH
München, Germany
mikael.va.vaaraniemi@bmw.de

Marc Treib Rüdiger Westermann
Technische Universität München
München, Germany
{treib,westermann}@tum.de

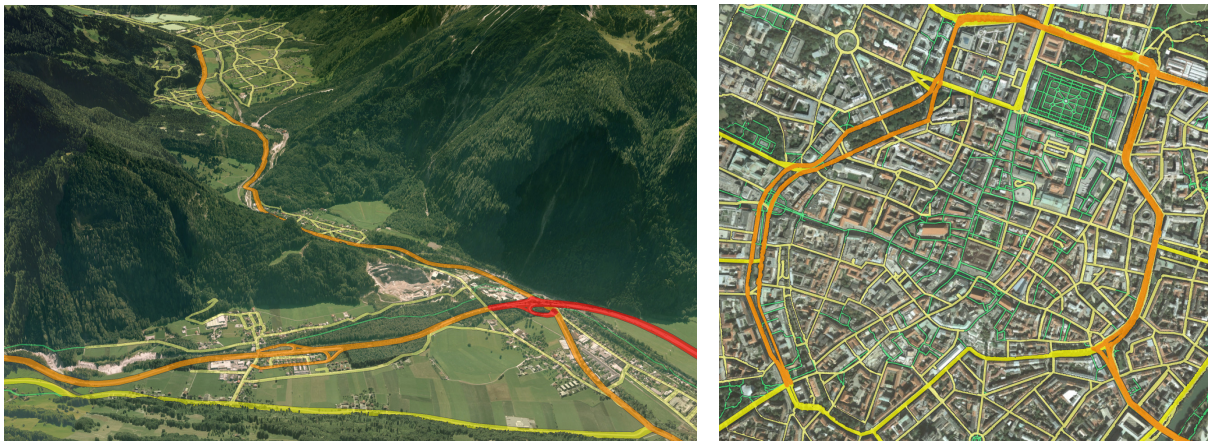


Figure 1: Cartographic rendering of roads in the Vorarlberg region, Austria, and in central Munich, Germany.

ABSTRACT

The efficient and high quality rendering of complex road networks—given as vector data—and high-resolution digital elevation models (DEMs) poses a significant problem in 3D geographic information systems. As in paper maps, a cartographic representation of roads with rounded caps and accentuated clearly distinguishable colors is desirable. On the other hand, advances in the technology of remote sensing have led to an explosion of the size and resolution of DEMs, making the integration of cartographic roads very challenging. In this work we investigate techniques for integrating such roads into a terrain renderer capable of handling high-resolution data sets. We evaluate the suitability of existing methods for draping vector data onto DEMs, and we adapt two methods for the rendering of cartographic roads by adding analytically computed rounded caps at the ends of road segments. We compare both approaches with respect to performance and quality, and we outline application areas in which either approach is preferable.

Keywords

cartography, vector draping, shadow volume, GIS, roads, terrain.

1. INTRODUCTION

Geographic Information Systems (GIS) store, analyze and visualize geo-referenced data. Road networks, land usage regions and selected points of interest are usually stored as vector data. In urban planning, cartography, and for navigation purposes, the visualization of roads on digital terrain models plays an important role [Döl05]. GIS engines should be able to handle and display such vector data efficiently and at high quality. A bare and uncluttered visualization as in paper maps is desirable. This *cartographic* representation of roads requires vivid colors, dark edges, rounded caps and runtime scaling of road width [Kra01, RMM95]. Dynamic scaling allows the perception of roads at every distance. In a cartographic rendering, roads are tinted using vivid colors to distinguish them from the underlying terrain. Associating different colors to each type of road induces an automatic cognitive grouping of similar roads [Kra01]. In addition, dark edges around roads add visual contrast [RMM95]. Examples of such cartographic representations are shown in Fig. 1 and 2. An additional aspect of a cartographic representation are rounded caps at each road segment. This avoids the ap-

Figure 2: Cartographic rendering of road maps using vivid colors and dark edges to achieve a high visual contrast to the underlying terrain.

pearance of cracks between segments and makes the visualization more appealing by introducing smooth endings and avoiding undesirable angular corners.

Another important information layer in GIS is the digital elevation model (DEM). It is usually given as raster data defining a 2.5D height map. Since the resolution and size of these DEMs are increasing rapidly, rendering approaches must be capable of dealing with TBs of data and gigantic sets of primitives that have to be displayed at high frame rates. To cope with these requirements, visualization techniques employ adaptive Level-Of-Detail (LOD) surface triangulations [LKR+96] and data compression, combined with sophisticated streaming and pre-fetching strategies [DSW09]. In such scenarios, the combined visualization of roads and a high-resolution DEM in a single visualization engine becomes a challenging task.

The main contribution of this paper is a method for rendering cartographic roads with rounded caps on high-resolution DEMs. We extend existing vector draping methods by introducing the possibility to compute caps analytically, thus avoiding an explicit triangulation. In this way we achieve a high-quality appearance without increasing the number of geometric primitives to be rendered. Furthermore, we introduce screen-space road outlines, runtime width scaling, and correct treatment of road intersections.

We have integrated our method into a tile-based terrain rendering engine. During preprocessing, this engine builds an multiresolution pyramid for both the DEM and the photo texture. It then partitions each level into square tiles, creating a quad tree. Each tile stores a Triangulated Irregular Network (TIN) representation of the DEM along with the photo texture. During runtime, tiles are chosen based on a maximum allowed screen-space error. In combination, this enables interactive 3D browsing of high-resolution terrain data with superimposed cartographic roads.

2. RELATED WORK

Terrain Rendering. Terrain rendering approaches using rasterization have been studied extensively over the last years. They employ the GPU to render large sets of polygonal primitives, and they differ mainly in the hierarchical height field representation used. There is a vast body of literature related to this field and a comprehensive review is beyond the scope of this paper. However, Pajarola and Gobbetti [PG07] discuss the basic principles underlying such techniques and provide many useful algorithmic and implementation-specific details. A thorough discussion of terrain rendering issues that are specifically related to high resolution fields is given in [DSW09].

Vector Data. The mapping of vector data on DEMs is an active research subject. The existing methods can be broadly classified into geometry-based, texture-based and shadow volume-based approaches.

Geometry-based methods generate and render separate primitives for the vector data. As the sampling frequency of the vector data generally does not match the triangulation of the underlying terrain, an adaption to the terrain triangulation and its LOD scheme is necessary. Because of this preprocess, geometry-based algorithms are strongly tied to the terrain rendering system and usually only allow static vector data [ARJ06, SGK05, WKW+03].

Texture-based techniques map the vector data onto a DEM in two steps: first, the data is rendered into off-screen textures either at runtime or in a preprocess. Afterwards, these textures are overlaid onto the terrain using texture mapping [DBH00]. This approach does not produce any aliasing artifacts thanks to hardware-supported texture filtering. Additionally, these methods are independent of the underlying terrain triangulation algorithms.
Static texturing methods provide high performance, but do not allow runtime changes of rendering parameters. Further problems occur at large zoom factors, as only limited resolution textures can be precomputed—there is an inherent tradeoff between the memory requirements and the obtainable quality [DBH00]. Therefore, Kersting and Döllner [KD02] combine this approach with on-demand texture pyramids: associating each quadtree region with an equally sized texture allows on-the-fly generation of appropriate textures. Dynamic vector data can be visualized if these textures

are created in each frame. However, this severely impacts performance, as many render target switches are needed. To overcome this, Schneider et al. [SGK05] introduce an approach using a single reparameterized texture for the vector data, analogously to perspective shadow mapping (PSM) [SD02]. As in PSM, some aliasing artifacts occur.

Bruneton and Neyret [BN08] present an approach that adapts the terrain heightfield to constraints imposed by the vector data (e.g. to enforce locally planar roads). Their method works only on regular meshes and would be difficult to generalize to our TIN-based terrain system. It is also not feasible for high-resolution terrain data. Additionally, an adaption of the heightfield is only necessary if the terrain resolution is insufficient to resolve such constraints, or if real-time editing is desired.

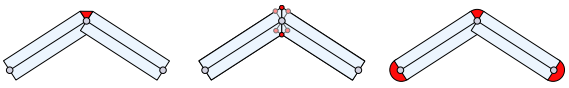
A **shadow volume-based** approach, recently introduced by Schneider and Klein [SK07], uses the stencil shadow volume algorithm to create pixel-exact draping of vector data onto terrain models. A stencil mask is created by extruding polygons along the nadir and computing the screen-space intersection between the created polyhedra and the terrain geometry. Using this mask, a single colored fullscreen quad is drawn. For each color, a separate stencil mask has to be generated. However, as the number of different vector data colors is typically small, this is not a major problem. The approach does not require any precomputations and is thus completely independent of the terrain rendering algorithm.

Our goal is to render *cartographic* roads on a *high-resolution* DEM. Continuous road scaling is a prerequisite, which makes texture-based approaches unsuitable. Likewise, a runtime triangulation of roads to match the DEM is not feasible, so most existing geometry-based approaches are not usable in our case.

We chose to use the shadow volume approach, as it does not require a preprocess and thus allows for runtime scaling of roads. It also provides pixel-exact projections. As a simpler and faster alternative, we also investigate a geometry-based approach where we adapt only the road centerlines to the DEM, so road scaling remains possible.

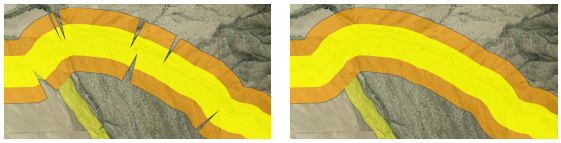
3. CARTOGRAPHIC ROADS

In GIS, roads are usually stored as vector data, i.e. as a collection of 2D polylines. One possibility to visualize such data is to convert the vector data into geometric primitives that are rendered on top of the terrain. However, a naive extrusion of each line segment to a quad results in the appearance of cracks between segments. The higher the curvature of a polyline, the more these cracks become visible. Two pragmatic solutions exist: filling the holes with additional triangles (see Fig. 3(a)) or connecting the corners of adjacent quads (see Fig. 3(b)). Both solutions are only possible if adjacency information is available. In real data sets, however, this information is commonly incomplete. Fig. 4 shows an example from a real data set where one continuous road is represented by several individual polylines, resulting in cracks between adjacent road segments where the polylines meet. We therefore choose a robust and elegant solution, which draws caps to avoid the appearance of cracks (see Fig. 3(c)) and does not require adjacency information. In addition to filling cracks, this approach generates visually pleasant smooth road endings (see Fig. 5, top). It also naturally handles sharp turns in a road (Fig. 5, bottom). Many major navigation systems visualize roads using rounded caps, for example Nokia with Ovi Maps, Google with Google Maps, Navigon and TomTom. It has become a de-facto standard technique when rendering cartographic roads [Phy09]. A naive method for rendering caps is the triangulation of a half-circle, leading to a large number of additional triangles per segment. Furthermore, the discrete triangulation becomes visible at large zoom factors. In the following sections, we present two methods that allow using perfectly round caps while avoiding an increase of the triangle count.



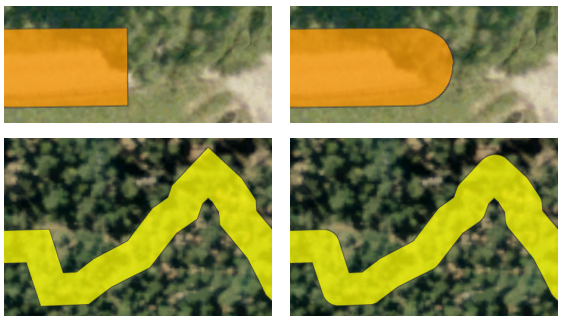
(a) Extra triangle (b) Moving corners (c) Rounded caps

Figure 3: Methods for removing cracks between quads.



(a) Cracks (b) Fixed with caps

Figure 4: Cracks occur because of missing adjacency information.



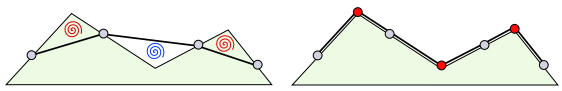
(a) No rounded caps (b) With rounded caps

Figure 5: Quality improvement with rounded caps.

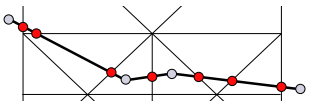
4. GEOMETRIC APPROACH

Our first method renders cartographic roads using a geometry-based approach. From the initial polyline

representation of a road, we individually process each line segment defined by successive vertices. In a preprocess, these lines are clipped against the terrain mesh in 2D, inserting additional vertices at each intersection (see Fig. 6). For more details on this preprocess, see section 6.1.



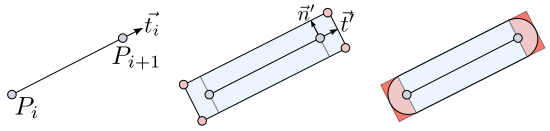
(a) Incorrect mapping of a road (grey vertices); problematic areas are marked by spirals (b) Correct mapping of the road using additional vertices (red)



(c) Top-down view with additional vertices

Figure 6: Geometry-based mapping of roads onto a terrain mesh.

To render rounded caps, we do not explicitly triangulate half-circles at the beginning and the end of each road segment. Instead, we render a single quad encompassing an entire road segment and evaluate the caps analytically in a shader program [Gum03] (see Fig. 7).



(a) Line segment i (b) Generated quad (c) Analytical caps

Figure 7: Analytical evaluation of rounded caps on a base quadrilateral.

We use the endpoints P_i and P_{i+1} of each line segment and the tangent $\vec{t}_i$ to generate a quad encompassing both capped ends (see Fig. 7(a) and (b)).

The caps are cut out of the generated quad in a pixel shader. We create a normalized local coordinate system inside both caps [RBE+06], which allows determining those fragments of a quad that are outside the cap and have to be discarded (see Fig. 7(c)).

Given points P_0, P_1, the ratio h between their distance $d = \overline{P_0P_1}$ and the cap radius $\frac{w}{2}$ is given by

$$h = \frac{w}{(d + 2 \cdot \frac{w}{2})} = \frac{w}{d+w}.$$

Equipped with h, we generate the local coordinates inside the caps with

$$x_{cap} = \frac{|x|-1}{h} + 1 \quad , \qquad y_{cap} = |y|.$$

If $x_{cap} > 0$, the fragment lies inside the cap area (the red area in Fig. 7(c)). If $x_{cap}^2 + y_{cap}^2 > 1.0$, it is outside of the half circle that builds the cap, and is discarded.

5. SHADOW VOLUME APPROACH

Our second algorithm is an extension of the shadow volume-based approach introduced by Schneider and Klein [SK07]. We extrude the road geometry along the nadir and apply a stencil shadow volume algorithm [Cro77, Hei91]. Thus, we compute the intersections between the extruded roads with the terrain geometry, resulting in per-pixel accurate projections onto the terrain. Similar to the approach described in section 4, we extend this algorithm by adding analytic rounded caps. We enlarge the geometry of each line segment to encompass the caps, and construct a local coordinate system that allows us to determine the fragments lying inside or outside the cap area. In the inside area, we analytically evaluate the caps via an intersection test between a ray and a cylinder and compute the depth value of the intersection point to be used during the depth test.

5.1. Intersection

From the camera position O, the fragment position F, and the view direction $\vec{v} = (F-O)/|F-O|$ we construct the view ray $R = O + t\vec{v}$. Given such a ray, the intersection of the ray with the cylinder spanned by the cap can be computed. Because the cylinder is always aligned with the z axis (the nadir), we can replace the 3D ray-cylinder test by a 2D ray-circle test in the xy plane (see Fig. 8).

A circle with center C and radius r is defined by the equation

$$(X-C)^2 = r^2.$$

Inserting the ray R into this equation with $\vec{c} := O - C$ yields

$$((O + t\vec{v}) - C)^2 = (\vec{c} + t\vec{v})^2 = r^2.$$

Expanding this results in the quadratic equation

$$(\vec{v}\cdot\vec{v})\, t^2 + 2\,(\vec{v}\cdot\vec{c})\, t + (\vec{c}\cdot\vec{c} - r^2) = 0.$$

Solving for t gives the discriminant

$$d = 4\,(\vec{v}\cdot\vec{c})^2 - 4\,(\vec{v}\cdot\vec{v})\,(\vec{c}\cdot\vec{c} - r^2).$$

If $d \leq 0$, there is none or only a single solution to the quadratic equation. This means that the ray does not hit the cap at all, or just grazes it. In this case, we discard the fragment. Otherwise, we get

$$t_{1/2} = \frac{-2\,(\vec{v}\cdot\vec{c}) \;\pm\; \sqrt{d}}{2\,(\vec{v}\cdot\vec{v})}.$$

For front faces, $\min(t_1, t_2)$ is the correct solution, for back faces it is $\max(t_1, t_2)$.

So far, we have assumed that the road geometry is extruded toward infinity to generate the shadow volumes. Since this is wasteful in terms of rasterization fill rate, we consider the height field for limiting the extent of

the shadow volumes. Assuming the terrain being partitioned into tiles, it is sufficient to extrude each line segment only within the extent of the bounding box of the tile it belongs to.
To accommodate this, the intersection algorithm has to be extended to handle the top and bottom sides of the extruded polyhedron: If the 2D distance between F and C is smaller than the cap radius (which can only happen for fragments belonging to the top or bottom side), F already gives the final intersection.

5.2. Numerical Precision

The algorithm as presented so far suffers from problems caused by limited numerical precision. One such problematic situation is depicted in Fig. 8: The intersection between each ray and the cylinder is computed twice, once for the front face of the bounding box (corresponding to F_0 in the figure) and once for the back face (corresponding to F_1). The ray direction is computed as $F_0 - O$ and $F_1 - O$, respectively. Because of small perturbations in F_0 and F_1, which are caused by the limited precision of the interpolation hardware, one of the intersection tests may report an intersection while the other one does not. This results in inconsistent output causing visible artifacts.

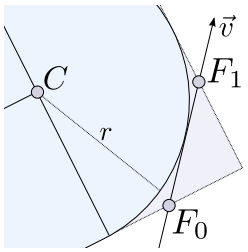


Figure 8: Numerically problematic ray-circle intersection.

In order to achieve consistent results, we compute both intersections in the same shader invocation: We render the geometry with front face culling enabled, and analytically compute the entry point into the bounding box of the extruded road. We then compute both intersections between the ray and the road as described above. This results in two depth values z_0, z_1 that need to be compared to the terrain depth z_t. We therefore replace the regular depth test with a custom two-sided test: z_t is read from a texture created as a secondary render target during the terrain rendering pass. If $z_0 < z_t < z_1$, then the road volume intersects the terrain geometry; otherwise, we discard the fragment.

Two beneficial side effects of this approach are that only half the amount of geometry needs to be rasterized compared to the naive approach, and that in contrast to the original shadow volume algorithm it does not require the rendering of full-screen quads to color the intersections.

6. IMPLEMENTATION DETAILS

In our proposed GIS engine, we visualize vector data e.g. from the OpenStreetMap project [Ope10]. Road networks are stored as a collection of polylines. Each polyline has a *functional road class* (FRC) [Tal96], defining a distinct width and color. For efficient data management at runtime, we partition the vector data into quadtree tiles, similar to the terrain data. Inside each tile, roads are stored sorted by their FRC.

6.1. Geometry Clipping

To avoid an incorrect mapping of roads onto the DEM in the geometric approach as in Fig. 6(a), we apply a preprocess where the centerline of each road segment is clipped against the terrain mesh in 2D. Additional vertices are inserted at each intersection (see Fig. 6(c)). However, finding the exit point of a line in a triangle by line-line intersection tests with the triangle edges provides poor numerical stability. We therefore perform these calculations in barycentric coordinates as illustrated in Fig. 9.

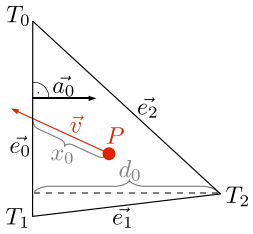


Figure 9: Computing line – triangle edge intersections.

We trace a line starting at point P along the normalized direction vector $\vec{v}$ in the triangle defined by the vertices T_0, T_1 and T_2. The change in the barycentric coordinate λ_2 of P with respect to T_2 is given by the signed distance moved along $\vec{a_0}$ divided by the distance d_0 of T_2 from $\vec{e_0}$, where $\vec{a_0}$ is a normalized vector perpendicular to $\vec{e_0}$ and pointing inside the triangle. When moving along $\vec{v}$, this becomes $(\vec{a_0} \cdot \vec{v})/d_0$. If this value is larger than zero, $\vec{v}$ is pointing away from $\vec{e_0}$ and we skip this edge. Otherwise, the maximum distance x_0 we can move along $\vec{v}$ before we hit $\vec{e_0}$ is given by

$$x_0 = \frac{\lambda_2 d_0}{\vec{a_0} \cdot \vec{v}}.$$

This can be done analogously for the other edges to compute x_1 and x_2; the smallest of these provides the actual exit point. At this point, an additional vertex is inserted into the polyline.

6.2. Cartographic Rendering

Scaling. In cartographic rendering, roads should be visible at all zoom levels. Therefore, while zooming out our system scales the roads' widths continuously.

The scaling factor is determined by the distance to the viewer. To avoid that roads close to the viewer become too wide, we only scale roads that are further away from the user than a given distance threshold (see Fig. 10).

(a) No scaling (b) Constant scaling (c) Distance-based scaling

Figure 10: Scaling of road width. Without scaling, distant roads become too narrow (left). A constant scale makes close roads too wide (middle). Distance-based width scaling gives satisfactory results (right).

Intersections. At crossroads or junctions, multiple roads of potentially different FRCs overlap, resulting in visible artifacts caused by additive blending. To resolve this problem, we draw roads into an offscreen render target without blending, in increasing order of importance.
The same approach allows for an easy integration of multi-colored roads by drawing a road multiple times with different widths and colors. This increases the geometry count proportionally to the number of colors, but since typically only a few important roads use multiple colors, this is acceptable. Fig. 11 demonstrates the correct handling of intersections of roads with different FRCs, including a two-color motorway.

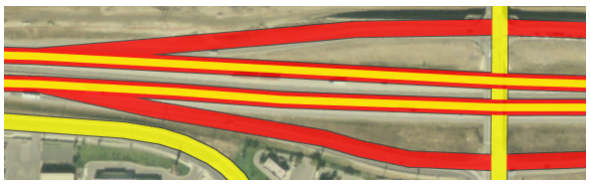


Figure 11: Correct handling of road intersections.

Outlines. To distinguish cartographic roads from the underlying terrain, we add dark edges around roads to increase contrast [RMM95]. To detect edges in screen space, we use a 3×3 or 5×5 kernel to find the local maximum road intensity $\alpha_{\max}$ around each fragment. The road intensity is the road opacity for pixels which are covered by a road, and 0 otherwise. The difference $\alpha_{\max} - \alpha_{\text{current}}$ defines the resulting edge intensity. Fig. 12 demonstrates the increase in visibility achieved by using outlines around roads.

(a) Without outlines (b) With outlines

Figure 12: Improving visibility by using dark outlines.

7. RESULTS

We have tested the proposed algorithms using three high-resolution data sets:

- A DEM of the US State of Utah, covering an area of about $276,000$ km^2 at a geometric resolution of 5 m. The road data set contains about $6,839,000$ vertices (216 MB).
- A DEM of Bavaria in Germany, covering an area of about $70,500$ km^2 at a geometric resolution of up to 80 cm. The road data set contains about $5,697,000$ vertices (151 MB).
- A DEM of the Vorarlberg region in Austria, covering an area of about $4,760$ km^2 at a geometric resolution of 1 m. The road data set contains about $213,000$ vertices (7 MB).

The size of the terrain data including photo textures is around 1 TB per data set. We therefore use an out-of-core visualization system capable of handling arbitrarily large data sets.
The preprocessing step for the geometric approach (see section 6.1) increased the size of the road data by about a factor of ten in all tested cases. Note that for the shadow volume approach, this step is not required.

Performance. All performance measurements were taken at a display resolution of 1600×1200 on a PC with Windows Vista, a 2.66 GHz Intel Core 2 Quad CPU, 8 GB of RAM and an ATI Radeon HD 5870 GPU (driver version 10.6).
The graph in Fig. 13 shows the frame rate during a recorded flight over the medium-resolution DEM of Utah at an average speed of about 1750 m/s. When rendering geometric roads (GEO), the maximum (average) performance drop is about 30% (26%) compared to rendering the terrain without roads. The highest performance impact occurs over Salt Lake City (far right in the graph). This area contains a dense road network and only a small amount of terrain geometry, as buildings are not included in the height field. The additional rendering of rounded caps does not significantly influence the performance.
For shadow volume-based roads (SV), the maximum (average) performance drop is around 40% (35%) without and 55% (42%) with rounded caps. Breaking the numbers down to the sole rendering of roads, SV with caps is about 1.4 times as expensive as without caps.

The visual quality produced by both techniques is identical at most locations in Utah. Therefore, GEO is preferable because of its higher performance. Fig. 14

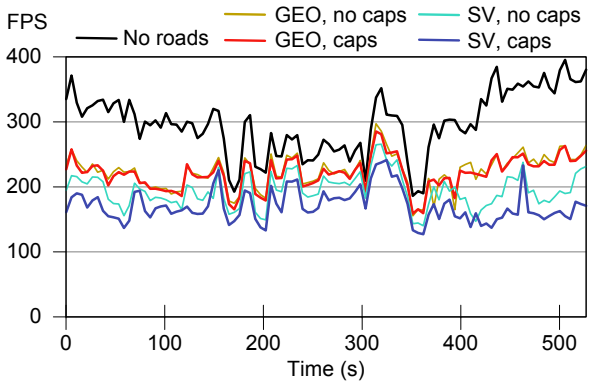


Figure 13: Performance - Utah

shows the frame rates during a flight over the high-resolution data set of Bavaria at an average speed of about 950 m/s. In this scenario, the performance of all approaches is very close; the average cost is between 33% and 43%. Even though GEO often requires many more triangles (up to about 3 million) than SV ($\leq$ 1M) because of the adaption to the terrain mesh (which itself uses up to about 35M triangles), GEO is still slightly faster. Thus, the GPU is more limited by shading computations than by the geometry throughput. However, GEO can often not provide an adequate mapping on such high-resolution terrain data (see Fig. 15). Therefore, SV is preferable for such fine-grained DEMs.

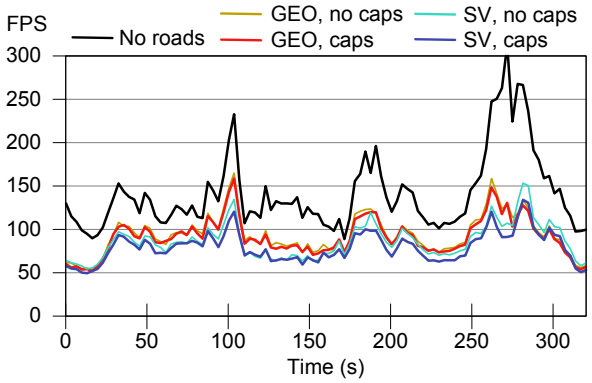
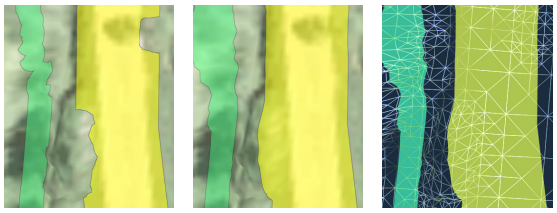


Figure 14: Performance - Bavaria

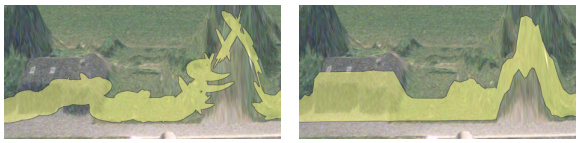


(a) Geometric (b) Shadow volume (c) Wire frame

Figure 15: Comparison of our draping algorithms on high-resolution terrain.

The Vorarlberg data set has a similar geometric resolution as the Bavaria data set, but the road network is much more sparse. Regardless of which algorithm is chosen for rendering roads, the highest performance impact amounts to only 15%. However, as in Bavaria, GEO can not provide adequate quality.

Matching. We should note that in many situations the vector data set did not exactly match the terrain data, i.e. there was a certain offset between the vector data roads and roads in the phototextures. These problems frequently occur in cities or forests, where even a slight offset causes a road to be projected onto a building or a tree. GEO fails to produce any reasonable results in this case (see Fig. 16(a)); SV produces a technically correct but not very useful projection (see Fig. 16(b)). This is a problem of the data rather than the draping algorithm. An additional preprocessing step could match the vector data to the terrain and its phototextures.



(a) Geometric (b) Shadow volume

Figure 16: Artifacts caused by a mismatch between terrain and vector data.

Comparison. Our method presents a marked improvement over several commercial GIS systems. For example, Google Earth 6.0 uses a simple geometric approach without adaption to the terrain and therefore does not achieve a correct projection of roads onto the DEM. It also does not provide correct road intersections and does not support multi-color roads or outlines. ArcGIS 10.0 rasterizes vector data into textures which are overlaid onto the terrain, similar to the orthophotos. This results in a correct projection and correct behavior at road intersections. However, a dynamic scaling of road widths is not possible, and multi-color roads or outlines are not supported.

8. CONCLUSION

In this paper, we have proposed and evaluated two approaches for rendering high-quality cartographic roads with rounded caps on high-resolution 3D terrain models. Both can be used on hardware platforms supporting Direct3D 10 or OpenGL 3.0. We have shown that a geometry-based approach provides high performance and good quality for low- to medium-resolution terrain data sets. However, it requires a moderately complex preprocessing step, and it can not provide an adequate visual quality with high-resolution terrain data. It is therefore a reasonable choice for low-end hardware, e.g. on mobile devices, where rendering of high-resolution terrain data is not feasible.

The shadow volume algorithm enables pixel-exact rendering of cartographic roads on 3D terrain. It is more expensive at runtime than the geometry-based approach; however, the rendering of high-resolution terrain remains the larger part. In low-resolution terrain data sets, on the other hand, its relative performance impact is large. The algorithm is easy to integrate into existing terrain rendering engines, as no adaption of roads to the terrain is required. It also extends naturally to polygonal vector data.

In further research, we plan to evaluate the use of tesselation shaders for the creation of geometric caps on Direct3D 11 or OpenGL 4.0 capable hardware.

9. ACKNOWLEDGEMENTS

The authors wish to thank the Landesvermessungsamt Feldkirch, Austria, the Landesamt für Vermessung und Geoinformation Bayern and the State of Utah for providing high-resolution geo data.
This publication is based on work supported by Award No. UK-C0020, made by King Abdullah University of Science and Technology (KAUST).

10. REFERENCES

[ARJ06] Agrawal, A., Radhakrishna, M., and Joshi, R. Geometry-based mapping and rendering of vector data over LOD phototextured 3D terrain models. In *Proceedings of WSCG*, pages 787–804, 2006.

[BN08] Bruneton, E. and Neyret, F. Real-time rendering and editing of vector-based terrains. In *Comput. Graph. Forum*, volume 27, pages 311–320, April 2008. Special Issue: Eurographics '08.

[Cro77] Crow, F. C. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.*, 11(2):242–248, 1977.

[DBH00] Döllner, J., Baumann, K., and Hinrichs, K. Texturing techniques for terrain visualization. In *VISUALIZATION '00: Proceedings of the 11th IEEE Visualization 2000 Conference (VIS 2000)*, Washington, DC, USA, 2000. IEEE Computer Society.

[Döl05] Döllner, J. Geovisualization and real-time 3d computer graphics. In E., Dykes, J., MacEachren, A., and Kraak, M., editors, *Exploring Geovisualization*, chapter 16, pages 325–343. Pergamon, 2005.

[DSW09] Dick, C., Schneider, J., and Westermann, R. Efficient geometry compression for GPU-based decoding in realtime terrain rendering. *Computer Graphics Forum*, 28(1):67–83, 2009.

[Gum03] Gumhold, S. Splatting illuminated ellipsoids with depth correction. In *Proceedings of 8th International Fall Workshop on Vision, Modelling and Visualization*, volume 2003, pages 245–252, 2003.

[Hei91] Heidmann, T. Real shadows, real time. *IRIS Universe*, 18:28–31, 1991.

[KD02] Kersting, O. and Döllner, J. Interactive 3d visualization of vector data in GIS. In *GIS '02: Proceedings of the 10th ACM international symposium on Advances in geographic information systems*, pages 107–112, New York, NY, USA, 2002. ACM.

[Kra01] Kraak, M. *Cartographic principles*. CRC, 2001.

[LKR+96] Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. Real-time, continuous level of detail rendering of height fields. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 109–118, New York, NY, USA, 1996. ACM.

[Ope10] OpenStreetMap. OpenStreetMap website, 2010.

[PG07] Pajarola, R. and Gobbetti, E. Survey of semi-regular multiresolution models for interactive terrain rendering. *Vis. Comput.*, 23(8):583–605, 2007.

[Phy09] Physical Storage Format Initiative. *Navigation Data Standard: Compiler Interoperability Specification*, 2009.

[RBE+06] Reina, G., Bidmon, K., Enders, F., Hastreiter, P., and Ertl, T. GPU-Based Hyperstreamlines for Diffusion Tensor Imaging. In *Proceedings of EUROGRAPHICS - IEEE VGTC Symposium on Visualization 2006*, pages 35–42, 2006.

[RMM95] Robinson, A., Morrison, J., and Muehrcke, P. *Elements of cartography*. John Wiley & Sons Inc, 1995.

[SD02] Stamminger, M. and Drettakis, G. Perspective shadow maps. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 557–562, New York, NY, USA, 2002. ACM.

[SGK05] Schneider, M., Guthe, M., and Klein, R. Real-time rendering of complex vector data on 3d terrain models. In Thwaites, H., editor, *The 11th International Conference on Virtual Systems and Multimedia (VSMM2005)*, pages 573–582. ARCHAEOLINGUA, October 2005.

[SK07] Schneider, M. and Klein, R. Efficient and accurate rendering of vector data on virtual landscapes. *Journal of WSCG*, 15(1-3), January 2007.

[Tal96] Talvitie, A. *Functional Classification of Roads*. Transportation Research Board, Washington, D.C., 1996.

[WKW+03] Wartell, Z., Kang, E., Wasilewski, T., Ribarsky, W., and Faust, N. Rendering vector data over global, multi-resolution 3d terrain. In *VISSYM '03: Proceedings of the symposium on Data visualisation 2003*, pages 213–222, Aire-la-Ville, Switzerland, 2003. Eurographics Association.