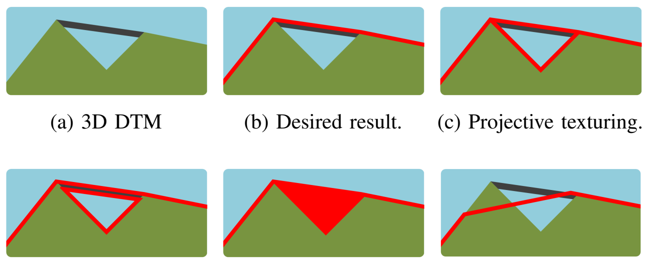
Real-time Screen-space Geometry Draping for 3D Digital Terrain Models

Matthias Trapp
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
matthias.trapp@hpi.de

Jürgen Döllner
Hasso Plattner Institute,
Digital Engineering Faculty,
University of Potsdam, Germany
juergen.doellner@hpi.de

***Abstract*—A fundamental task in 3D geovisualization and GIS applications is the visualization of vector data that can represent features such as transportation networks or land use coverage. Mapping or draping vector data represented by geometric primitives (e.g., polylines or polygons) to 3D digital elevation or 3D digital terrain models is a challenging task. We present an interactive GPU-based approach that performs geometry-based draping of vector data on per-frame basis using an image-based representation of a 3D digital elevation or terrain model only.**

***Keywords*-Geometry Draping, Geovisualization, GPU-based Real-time Rendering**



(a) 3D DTM. (b) Desired result. (c) Projective texturing.

(d) Stencil based. (e) Shadow volumes. (f) Geometry draping.

Figure 1: Comparison of draping approaches.

I. Introduction

A. Motivation

Draping has a number of applications in geovisualization. Most prominent are the mapping of Geographic Information System (GIS) features such as transportation networks or planar polygon features. There are basically two types of draping primarily used: *Texture-based Draping (TBD)* denotes a perspective or panoramic rendering of a 2D texture superimposed onto a 3D surface. For example, an aerial photograph might be draped over a Digital Elevation Model (DEM) to facilitate terrain visualization.

In contrast thereto, *Geometry-based Draping (GBD)* denotes the process of fitting a given 2D point, line, or polygonal geometry, the *Draping Source (DS)* onto a 3D surface, denoted as the *Draping Target (DT)*. A DS can be represented by a 2D mesh, a set of 2D lines, or 2D point data with additional appearance variables such as color or texture. We extend the classical notion of draping sources by considering 2.5D representations.

B. Limitations of Existing Approaches

Previous work describes basically three different classes of approaches and their combinations for draping feature geometry on top of a DEM (cf. Figure 1). Table I compare these classes with respect to their ability to convey the characteristics of the DS. None of the existing approaches is able to convey all.

Geometry-based Draping (GBD): Rendering a poly-line on terrain by sub-sampling points along the poly-line requires adjustment in response to terrain Level-of-Detail (LOD) switches and the coplanar geometry can lead to z-fighting [1]. Thus, it often requires additional tessellation of the DS, which impacts the runtime performance depending on geometric complexity of the DS. Therefore, on-demand simplification (run-time overhead) or multi-scale representation (memory overhead) approaches can be used.

Texture-based Draping (TBD): This common approach for polygon draping requires the creation of a rasterized polygon representation (texture) and its subsequent terrain rendering using multi-texturing [2]. It avoids the complexity and drawbacks of draping geometry, with no tessellation required, no polygon-offsets, and no z-buffer concerns. Creating the texture in a pre-processing step limits the texture resolution to the available Video Random Access Memory (RAM) (VRAM), leading to aliasing as the viewer zooms in; generating the texture on-demand [3] can become expensive and does not prevent aliasing on steep slopes or blurry feature edges.

Stencil-based Draping (SBD): This rendering approach is based on a intermediate geometric representation (shadow volume) computed from the draping source. Thereby, input primitives (lines and triangle) are extruded in a pre-processing step (or during rendering using geometry shader functionality) and then rasterized with stencil buffering enabled [2]. Using subsequent multi-pass rendering with stencil testing, the DT can be colored according to the stencil buffer content.

Table I: Overview of existing draping approaches and how they preserve the different aspects of draping sources.

Aspect of DS to Preserve	GBD	TBD	SBD
Color	✓	✓	✓
Texture	✓	✓	✗
Geometry	✓	✗	✗
Height	✗	✗	✗

Rendering multiple DSs requires multiple intermediate representations that must be recompute if the DT is changed or modified. Using shadow volumes leads to smearing on steep slopes and dashing when viewed lengthwise [4].

Recent advances in Graphics Processing Unit (GPU)-based tessellation enables real-time GBD for dynamic DS *and* DT. Further, it supports the application of adaptive hardware-accelerated terrain tessellation in combination with dynamic DS, e.g., for the rendering and visualization of moving objects over dynamic surfaces.

C. Problem Statement

Existing approaches require a re-computation and update step if DT and DS are of dynamic nature, i.e., comprising animated geometry based on data set that are changing temporarily. That can include the support of LOD rendering of 3D Digital Terrain Model (DTM) at different tessellation levels. To enable high-quality draping and rendering of dynamic DSs and DSs, it is feasible to implement this using an image-based representation of the draping target, e.g., the rasterized 3D DTM [5]. Here, the basic research question is how to find the correspondence between the image-based representation of a digital 3D DEM or DTM and the geometric of the features that should be mapped in real-time. This question becomes more immanent if LOD rendering is considered, i.e., the correspondence must be computed for different geometric representations. Further, it can desirable to re-use already draped geometry (draped source) later on in subsequent processing and rendering stages, for example within GIS or 3D geovirtual environments.

D. Approach & Contributions

This paper presents a novel interactive approach to the challenges stated above. The real-time rendering technique is based on an image-based representation of DTs (G-Buffer) [5] or its extensions (e.g., K-Buffer [6]) to effectively decouple the rendering of the DT and DS geometry. Instead of computing intermediate data representations, our approach projects each vertex of the input geometry onto the image-based representation using a binary search algorithm []. To achieve sufficient vertex density, the tessellation and subdivision of a DS can be refined adaptively using hardware-accelerated techniques [].

The presented approach has a number of advantages. Since it requires only G-Buffer information, it is (1) independent of the specific rendering technique for the draping target (e.g., terrain renderer or similar), and thus (2) it is not require to respect underlying implications LOD rendering techniques and (3) is suitable multi-scale and view-dependent approaches. The draping process is based on geometric computation on graphics hardware and thus features a lower memory footprint compared to pre-processed geometry-based approaches. This basically allows to store intermediate or the final results for further processing.

Table II: Overview and classification of related work with respect to draping approaches. Hybrid approaches are highlighted by gray rows.

Approach	GBD	TBD	SBD
Kersting & Döllner [3]		●	
Wartell *et al.* [1]	●		
Schneider *et al.* [7]	●	●	
Agrawal *et al.* [8]	●		
Sun *et al.* [9], Deng *et al.* [10]	●		
Schneider & Klein [11], Dai *et al.* [12]			●
Schilling *et al.* [13]	●		
Sun *et al.* [14], Wang *et al.* [15]	●	●	
Yang *et al.* [16]		●	●
Vaaraniemi *et al.* [17]	●		●

To summarize, this paper presents the following contributions: (1) it presents a novel rendering technique for draping polygonal geometry onto virtual terrain models, and (2) it allows for decoupling the rendering of digital surface models from feature rendering.

The presented approach has various applications for geo-visualization of spatial-temporal feature data. The major use-case represents the mapping of features geometry to 3D DEM or DTM. It further can be used to complement approaches for rendering of transportation networks to mountainous regions. An additional application scenario lies in the domain of Computational Geometry: the computation and visualization of distances between possible dynamic polygons. In such application the distance between DS vertices and the DT is computed using the vertex projection approach and then visualized on the DS surface, e.g., using color coding.

The remainder of this paper is structured as follows. Section II reviews and discusses related work w.r.t. interactive geometry draping and 3D DTM. Section III introduces the concept of image-based geometry draping and describes implementation details of an prototypical interactive rendering technique and evaluates it run-time performance. Finally, Section IV concludes the paper and presents ideas for future research.

II. Related Work

Despite computing the draped polygonal representation on using a pre-processing step [13] or using screen-space approaches that are limited to line geometry [4] only, there are basically two major GPU-based approaches to the problem of mapping polygonal geometric data to terrain models: *texture-based* and *stencil-based* using stencil buffers. For the sake of completeness we also review *geometry-based* draping based on Central Processing Unit (CPU). Table II summarized major characteristics of the approaches reviewed in the following.

Geometry-based Draping (GBD): This class of approaches directly modify the vector data to fit the surface of the terrain representation and render the results using separate geometric primitives. Wartell *et al.* show how to overlay 2D poly-lines on top of terrain models [1]. The overlay polylines are rendered independently from other image data due to rasterization artifacts. They present the triangle clipping Directed Acyclic Graph (DAG) data structure, which allows rendering the projected polylines together with a quad-tree based terrain model. They address the challenging problem of combining progressive terrain meshes, which change at nearly every frame as described by e.g., Hoppe 1998 or Lindstrom and Pascucci 2002, with 3D polyline data, which also needs to adapt accordingly. Agrawal *et al.* use a similar technique for combining a textured terrain model with polyline data [8]. In this approach, the terrain is organized as block-based LOD structure derived from a height raster, which allows for efficient memory paging and using optimized data structures such as triangle-strips. Due to this block-based simplification and visualization scheme (nine tiles are visible at one time), height values can be sampled up for each line segment from the underlying mesh with the highest resolution. Over meshes with lower resolutions, these height values must be corrected accordingly.

Texture-based Draping (TBD): This approach rasterizes vector graphics into an texture to subsequently project it on the virtual 3D terrain model using hardware-accelerated projective texturing [18], [19]. Here, raster-based GIS layers can be overlaid and combined to a single texture layer. Additionally, hardware-accelerated techniques such as mipmap filtering can be used to optimize rendering speed and memory consumption [20]. There are basically two classes of this approach: *static* and *dynamic* TBD. Static TBD approaches create static multi-resolution texture pyramids in a pre-processing step. This requires large amounts of VRAM and out-of-core techniques for texture handling as well as suffers of sampling artifacts due to lack of texture resolution [4]. To overcome these limitations, Kersting and Döllner proposed on-demand texture pyramids [3]. In contrast to static method, texture pyramids are created for arbitrary resolutions on a per-frame basis using render-to-texture capabilities of a hardware rasterizer. The advantage of this method is that large amounts of texture data that would be necessary for every possible resolution do not need to be computed.

Hong *et al.* present an approach that renders vector data over 3D terrains using view-dependent perspective texture-mapping [21]. Since it is based on texturing only, the major advantage of using TBD is its independent of LOD-Rendering algorithms for DTs. However, the view-dependent pre-processing of texture data requires VRAM and can not guarantee high visual quality.

Stencil-based Draping (SBD): Schneider and Klein proposes a solution which extrudes the vector data to polyhedral and uses these to create a stencil buffer mask [11], similar to shadow volumes [2]. It is algorithmically complex, but avoid the limitations of conventional geometry and texture-based approaches for feature draping [12], since its independent from target geometry. Vaaraniemi *et al.* adapted this techniques for the rendering high-quality cartographic roads on high-resolution DEMs [17]. Due to the intermediate shadow-volume representation, the major limitations of SBD are: (1) re-computation of stencil volumes is required for dynamic DSs. Stencil volumes can be computed on CPU, which can become costly for draping targets of high geometric complexity, or using geometry shader [17]; (2) limited control of target appearance, especially texturing and shading, and (3) specific interaction techniques are required.

Hybrid Draping Methods: Further, research presented hybrid methods combining some or all of the above methods for the sake of rendering quality and rendering performance by benefiting from the advantages of both techniques. For example, Schneider *et al.* combine texture-based and geometry-based approaches for real-time rendering of geometrical complex vector data on 3D terrain models [7]. Further, the texture-based approach is extended using a perspective re-parametrization similar to that applied in perspective shadow mapping [22], yielding quality superior to standard texture mapping, while an geometry-based approach is used for high-quality visualizations. The approach is basically limited to static draping targets, because the 3D geometry is created from the 2D vector data for each LOD of the terrain and completely incorporated into the terrain quad-tree hierarchy of the LOD in a pre-processing step. A similar approach is used by Sun *et al.* for the interactive manipulation and display of large-scale vector data in 3D landscape maps [14], [15]. Further, Yang *et al.* combine geometry-based and stencil-based draping for an efficient rendering for large vector data on large terrain models [16].

III. Real-time Screen-space Geometry Draping

This section describes details of our prototypical implementation based on Open Graphics Library (OpenGL) [23] and OpenGL Shading Language (GLSL) [24]. Figure 2 shows an overview of draping visualization pipeline of our approach. It mainly comprises the three stages described in the remainder of the section (Sections III-A to III-C).

A. Terrain Rendering and G-Buffer Generation

This stage creates an image-based representation of the 3D DTM geometry. Such G-Buffer [5] can be generated within a single rendering pass using forward rendering with Render-To-Texture (RTT) functionality in combination with Multiple-Render-Targets (MRTs). It basically generates a color buffer

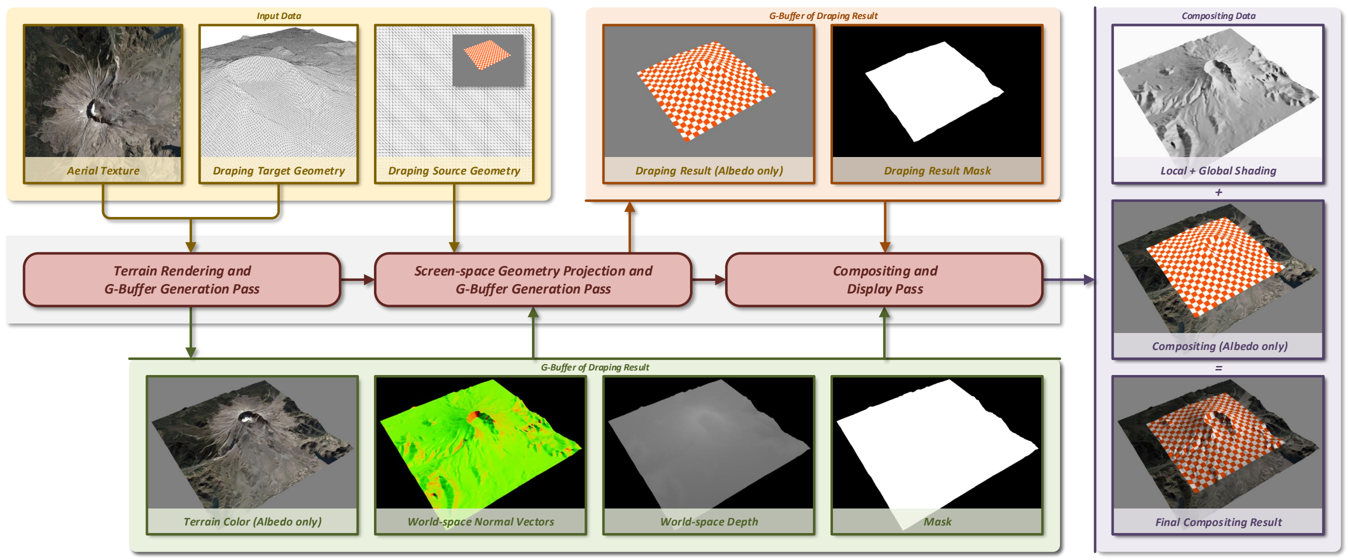

Input Data
Aerial Texture
Draping Target Geometry
Draping Source Geometry
G-Buffer of Draping Result
Draping Result (Albedo only)
Draping Result Mask
Compositing Data
Local + Global Shading
+
Terrain Rendering and G-Buffer Generation Pass
Screen-space Geometry Projection and G-Buffer Generation Pass
Compositing and Display Pass
Compositing (Albedo only)
=
G-Buffer of Draping Result
Terrain Color (Albedo only)
World-space Normal Vectors
World-space Depth
Mask
Final Compositing Result

Figure 2: Schematic Overview of data and processing components for screen-space geometry draping pipeline.

storing the terrain albedo color, a normal buffer storing *world-space normal vectors* for later shading, as well as an *position buffer* (floating point precision required), which is required by the geometry projection stage. Thus, this approach makes no assumptions of the on the actual terrain rendering algorithm used, e.g., it supports LOD and multi-scale rendering techniques.

The screen-space approach assumes the following image-based data [5] acquired from a terrain renderer: (1) albedo or color of textured terrain (Red-Green-Blue (RGB)), (2) surface normal N, and (3) the position in P in world-space coordinates. These buffers can be generated in single-rendering pass using MRT. For a more compact representation, normal in eye-space (x, y component only), and world-space position (z_P component only) can be stored together with the RGB using 2D texture arrays for an effective access during runtime. This representation can also be used for deferred shading or stylization (Section III-C).

B. *Screen-space Geometry Projection*

This section describes the basic components of draping polygonal geometry by only using G-Buffer information of the DT. For simple integration of the proposed approach, it is assumed that the DS is represented using geometric rendering primitives, i.e., points, lines, or triangles. The main task is basically projecting the vertices of these primitives to the image-based representation of the DT on a per-vertex basis using vertex shader. Figure 3 shows an overview of the projection concept. The basic idea is to find the intersection of a line L through V with the image-based DT representation obtained by the previous stage. Listing 1 shows a GLSL code for performing a binary search for such an intersection point. It takes the world-space normal and depth texture as well as a number of samples used together with a sampling

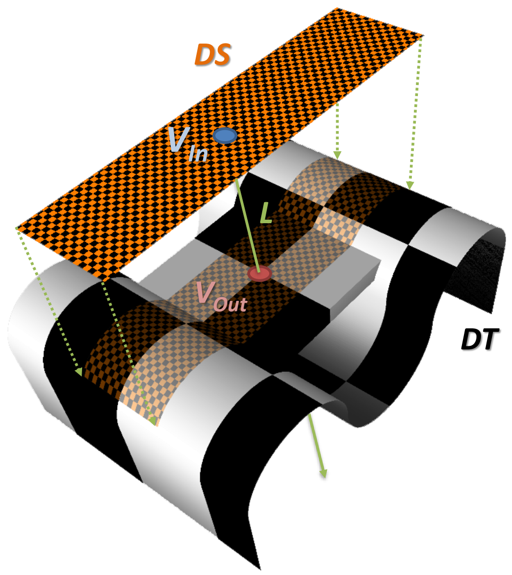


Figure 3: The position of the output vertex V_{Out} on the draping target (DT) is the result of draping the respective input position V_{in} of the draping source (DS) by performing an binary depth-intersection search along a line L.

step size required for averaging as input. The matrix `MVP` represents the model-view-projection transformation used for rendering [2].

C. *Image-based Compositing in Screen-space*

This final rendering pass combines the results of both previous rendering passes into a single output image. Further, deferred shading and lighting is applied based on the obtained G-Buffer data from the 3D DTM and the projected DS.

This step produces the final image by basically performing compositing based on the G-Buffer information of the 3D

```
// Convert to normalized screen coordinates
vec4 toNSC(const in vec4 v) {
 return vec4(0.5 * (v.xyz / v.w) + 0.5, v.w);
}

vec4 vertexDraping(
 const in sampler2D positionTex, // Position G-Buffer
 const in vec4 Vin, // Vertex to drape
 const in int samples, // #Samples for smoothing
 const in float sampleStep) // distance for smoothing
{
 float texSize = float(textureSize(positionTex, 0).x);
 float pixelSize = 1.0 / texSize;
 vec2  stepSize = vec2(sampleStep/texSize);
 // Construct line to search along
 vec4 lineStart = MVP * vec4(Vin.xy, 1.0, 1.0);
 vec4 lineEnd   = MVP * vec4(Vin.xy,-1.0, 1.0);
 vec4 Vout = Vin;
 // Binary search for line-terrain intersection
 float first = 0.0, last = 1.0;
 while(first <= last)
 {
  // Compute mid-point
  float mid = first + (last-first) / 2.0;
  // Compute texture coordinates along line
  vec4 texCoords = toNSC(mix(lineStart, lineEnd, mid));
  vec4 sample = vec4(0.0); // Sample terrain
  for(int s = -samples; s < samples; s++) {
   for(int t = -samples; t < samples; t++) {
    sample += texture(positionTex,
                 texCoords.st + vec2(s,t) * stepSize);
   }
  }
  // Smooth samples obtain from G-Buffer
  sample =  sample / float(samples * samples * 4.0);
  // Compute height from normalized representation
  float terrainHeight = (2.0 * sample.z - 1.0);
  Vout = vec4(Vin.xy, terrainHeight, 1.0);

  if((last-first) < pixelSize) // Termination criteria
   return Vout;
  // Perform intersection test
  float depthScene = toNSC(MVP * Vout).z;
  if(depthScene >= texCoords.z)
   first = mid;
  else
   last  = mid;
 }
 return Vout;
}
```

Listing 1: GLSL implementation of draping algorithm.

virtual terrain and the G-Buffer information of the projected feature. By sampling the G-Buffers at the respective fragment positions, the compositing is basically performed by using α-blending. This approach facilitates integration with modern deferred renderer and allows for feed-back to G-Buffer input for multiple draping.

D. Performance Evaluation

We tested the rendering performance of our prototypical implementation using a 3D DEM (DT) represented by a regular grid of 67 330 vertices and 131 072 triangle primitives (indexed). The performance test was conducted using a NVIDIA GeForce GTX 970 GPU with 4 096 MB VRAM on a Intel Xeon CPU with 2.8 GHz and 12 GB RAM rendering at a viewport resolution of 1280×720 pixels. The application runs in windowed with vertical synchronization turned off.

Table III: Rendering performance results for draping sources different geometric complexities.

# Vertices	# Triangles	Frames-per-Second (FPS)
2 500	4 800	310.6
4 900	9 522	226.3
10 000	19 602	152.1

The run-time performance mainly depends on the geometric complexity of the 3D scene and decreases w.r.t. the geometric complexity and number of DSs. Table III shows the obtained measurements in Frames-per-Second (FPS), averaged over 500 frames for DSs of different geometric complexity (also in indexed representation).

IV. Conclusions and Future Work

This paper presents a novel hardware-accelerated interactive visualization technique for draping vector features to 3D digital terrain models in screen space. For future work, the presented approach can be complemented with adaptive hardware-accelerated subdivision of sparse feature geometry to ensure sufficient vertex density.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group “4DnDVis”

References

[1] Z. Wartell, E. Kang, T. Wasilewski, W. Ribarsky, and N. Faust, “Rendering Vector Data over Global, Multi-resolution 3D Terrain,” in *Proceedings of the Symposium on Data Visualisation 2003*, ser. VISSYM ’03. Aire-la-Ville, Switzerland, Switzerland: Eurographics Association, 2003, pp. 213–222.

[2] T. Akenine-Möller, E. Haines, N. Hoffman, A. Pesce, M. Iwanicki, and S. Hillaire, *Real-Time Rendering 4th Edition*. Boca Raton, FL, USA: A K Peters/CRC Press, 2018.

[3] O. Kersting and J. Döllner, “Interactive 3D Visualization of Vector Data in GIS,” in *Proceedings of the 10th ACM International Symposium on Advances in Geographic Information Systems*, ser. GIS ’02. New York, NY, USA: ACM, 2002, pp. 107–112.

[4] D. Ohlarik and P. Cozzi, “A screen-space approach to rendering polylines on terrain,” in *ACM SIGGRAPH 2011 Posters*, ser. SIGGRAPH ’11. New York, NY, USA: ACM, 2011, pp. 68:1–68:1.

[5] T. Saito and T. Takahashi, “Comprehensible Rendering of 3-D Shapes,” in *Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH ’90. New York, NY, USA: ACM, 1990, pp. 197–206.

[6] L. Bavoil, S. P. Callahan, A. Lefohn, J. a. L. D. Comba, and C. T. Silva, "Multi-fragment Effects on the GPU Using the K-buffer," in *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games*, ser. I3D '07. New York, NY, USA: ACM, 2007, pp. 97–104.

[7] M. Schneider, M. Guthe, and R. Klein, "Real-time Rendering of Complex Vector Data on 3D Terrain Models," in *The 11th International Conference on Virtual Systems and Multimedia (VSMM2005)*, H. Thwaites, Ed. ARCHAEOLINGUA, Oct. 2005, pp. 573–582.

[8] A. Agrawal, M. Radhakrishna, and R. Joshi, "Geometry-based Mapping and Rendering of Vector Data over LOD Phototextured 3D Terrain Models," in *Short Communications Proceedings of WSCG 2006*, V. Skala, Ed. Plzen, Czech Republic: UNION Agency-Science Press, Jan. 2006.

[9] M. Sun, R. Zhao, J. Hu, and H. Guo, "3D Visualization Method of Large-Scale Vector Data for Operation," in *Cooperative Design, Visualization, and Engineering*, ser. Lecture Notes in Computer Science, Y. Luo, Ed. Springer Berlin Heidelberg, 2007, vol. 4674, pp. 257–260.

[10] B. Deng, D. Xu, J. Zhang, and C. Song, "Visualization of vector data on global scale terrain," in *Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering (ICCSEE 2013)*. Atlantis Press, Paris, France, mar 2013, pp. 85–88.

[11] M. Schneider and R. Klein, "Efficient and Accurate Rendering of Vector Data on Virtual Landscapes," *Journal of WSCG*, vol. 15, no. 1-3, pp. 59–64, Jan. 2007.

[12] C. Dai, Y. Zhang, and J. Yang, "Rendering 3D Vector Data using the Theory of Stencil Shadow Volumes," in *XXI Congress of the International Society for Photogrammetry and Remote Sensing*, W. K. Jun Chen, Jie Jiang, Ed., vol. XXXVII, Part B2. International Society for Photogrammetry and Remote Sensing, jul 2008.

[13] A. Schilling, J. Basanow, and A. Zipf, "Vector based Mapping of Polygons on Irregular Terrain Meshes for Web 3D Map Services," in *WEBIST 2007 - Proceedings of the Third International Conference on Web Information Systems and Technologies, Volume WIA, Barcelona, Spain, March 3-6, 2007.*, 2007, pp. 198–205.

[14] M. Sun, G. Lv, and C. Lei, "Large-scale Vector Data Displaying for Interactive Manipulation in 3D Landscape Map," in *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 37, 2008, pp. 507–512.

[15] X. Wang, J. Liu, and J. Bi, "Rendering of Vector Data on 3D Virtual Landscapes," in *Information Science and Engineering (ICISE), 2009 1st International Conference on*, Dec 2009, pp. 2125–2128.

[16] L. Yang, L. Zhang, Z. Kang, Z. Xiao, J. Peng, X. Zhang, and L. Liu, "An Efficient Rendering Method for Large Vector Data on Large Terrain Models," *SCIENCE CHINA Information Sciences*, vol. 53, no. 6, pp. 1122–1129, 2010.

[17] M. Vaaraniemi, M. Treib, and R. Westermann, "High-Quality Cartographic Roads on High-Resolution DEMs," *Journal of WSCG*, vol. 19, no. 2, pp. 41–48, 2011.

[18] P. Haeberli and M. Segal, "Texture Mapping As A Fundamental Drawing Primitive," in *Fourth Eurographics Workshop on Rendering*, M. F. Cohen, C. Puech, and F. Sillion, Eds., 1993, pp. 259–266.

[19] C. Everitt, "Projective Texture Mapping," NVIDIA Corporation, Tech. Rep., Apr. 2001.

[20] M. Trapp and J. Döllner, "Dynamic Mapping of Raster-Data for 3D Geovirtual Environments," in *13th International Conference on IEEE Information Visualisation*. IEEE Computer Society Press, 2009, pp. 387–392.

[21] C. Hong, T. Xiaoan, X. Yaohua, and S. Maoyin, "Rendering Vector Data over 3D Terrain with View-Dependent Perspective Texture-Mapping," *Journal of Computer-Aided Design & Computer Graphics*, vol. 22, no. 5, jan 2010.

[22] M. Stamminger and G. Drettakis, "Perspective Shadow Maps," *ACM Trans. Graph.*, vol. 21, no. 3, pp. 557–562, Jul. 2002.

[23] M. Segal and K. Akeley, *The OpenGL Graphics System: A Specification (Version 3.2 (Core Profile))*, The Khronos Group Inc., Jul. 2009.

[24] J. Kessenich, *The OpenGL Shading Language Language Version: 1.50 Document Revision 9*, The Khronos Group Inc., Jul. 2009.