Multiscale Visualization of 3D Geovirtual Environments Using View-Dependent Multi-Perspective Views

Sebastian Pasewaldt Matthias Trapp Jürgen Döllner
Hasso-Plattner-Institut, University of Potsdam, Germany
{sebastian.pasewaldt|matthias.trapp|juergen.doellner}@hpi.uni-potsdam.de

ABSTRACT

3D geovirtual environments (GeoVEs), such as virtual 3D city models or landscape models, are essential visualization tools for effectively communicating complex spatial information. In this paper, we discuss how these environments can be visualized using multi-perspective projections [10, 13] based on view-dependent global deformations. Multi-perspective projections enable 3D visualization similar to panoramic maps, increasing overview and information density in depictions of 3D GeoVEs. To make multi-perspective views an effective medium, they must adjust to the orientation of the virtual camera controlled by the user and constrained by the environment. Thus, changing multi-perspective camera configurations typically require the user to manually adapt the global deformation — an error prone, non-intuitive, and often time-consuming task. Our main contribution comprises a concept for the automatic and view-dependent interpolation of different global deformation preset configurations (Fig. 1). Applications and systems that implement such view-dependent global deformations, allow users to smoothly and steadily interact with and navigate through multi-perspective 3D GeoVEs.

Keywords: multi-perspective views, view-dependence, global space deformation, realtime rendering, virtual 3D environments, geovisualization.

1 INTRODUCTION

3D GeoVEs, such as virtual 3D city and landscape models, represent efficient tools for fields such as geography or cartography, in particular if their visualization and knowledge can be transferred to the 3D visualization domain [9]. Previous work has shown that global deformation applied to such environments can be used to assist wayfinding and navigation by making effective use of the available image space [10, 13] and by reducing occlusions [18]. Grabler et al. [2009] demonstrate that the usage of multi-perspective views in combination with cartographic generalization techniques such as simplification and deformation is suitable to convey important information with in 3D tourist maps.

In the context of interactive global deformations and multi-perspective views, existing visualization techniques and systems are most effective for specific settings of a virtual camera, i.e., Fig. 2. The virtual camera must be near the ground (pedestrian view) or at a certain height (birds-eye view), in order to exploit the full potential of these visualization techniques. Usually, in a 3D GeoVE the user wants to interact and navigate freely. This would require the manual adaptation of the visualization parameters during interaction and navigation. In general, this task is complex, error-prone, and time-consuming. In this paper we develop a concept that delivers a suitable visualization for a camera setting via automatic view-dependent interpolation of global deformations that are represented by parametric curves.

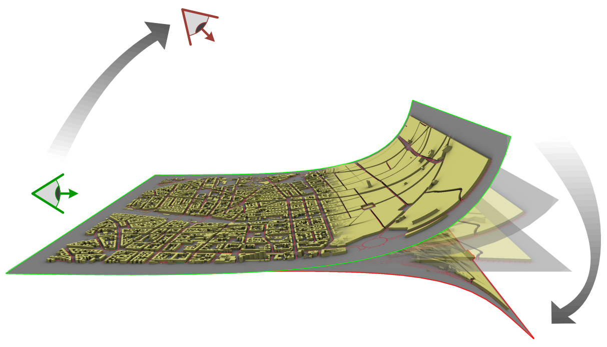

Figure 1: Conceptional sketch of the interpolation of global deformations and different geometric representations based on the viewing angle of the virtual camera.

Further, a drawback of 3D GeoVEs are the multiple geometric scales [9], introduced by the perspective projection of the camera, because they lead to small scales in the more distant parts of the scene. Consequently, the depiction of objects only have limited image space (e.g. only one pixel) and cannot be distinguished by a viewer (pixel noise). To overcome this problem in the domain of paper maps, cartographers apply generalization techniques to minimize visual complexity and to improve comprehension. A similar concept is used in most of the current multi-perspective techniques. Instead of using a photo-realistic style, a map-based style is applied to regions of small scales. We generalize the style concept

Please note, that this is a draft version. The final paper has been published as part of the Journal of WSCG 2011, volume 19, number 3, page 111-118.

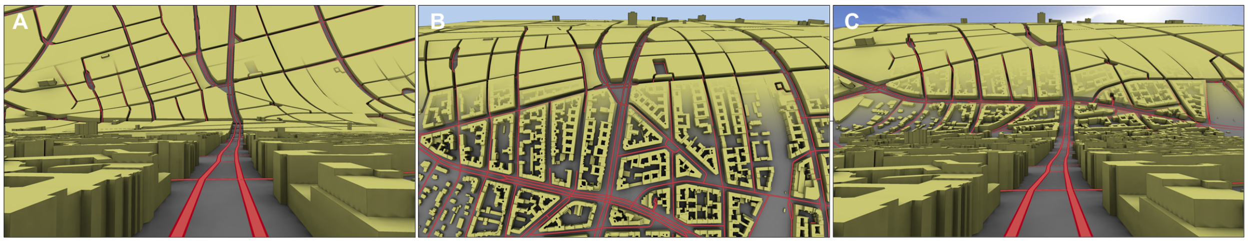

Figure 2: Exemplary results of our visualization system that enables the view-dependent interpolation of the depicted scenes: progressive perspective (A), degressive perspective (B), and a hybrid perspective (C) using different generalization levels of a 3D virtual city model of Berlin.

by letting the user define multiple geometric representations, e.g., obtained from cell-based generalization [8], to sections of the curve (Fig. 2). Further, these explicit geometric representations enable more design freedom then automatically derived style variations such as in [10]. Jobst and Döllner (2008) further suggest to subdivide the visualization into zones where a constant scaling and thus a constant generalization is applied per zone. An exemplary visualization can be seen in Fig. 7.

Möser et al. [13] generalize the concept introduced in [10] by using Hermite curves for the parameterization of global deformations, which can be easily manipulated by the user. However, the application of standard parameterized curves for such a visualization introduces additional geometric distortions. We compensate these by an arc-length parameterization [14].

In this work we present a concept and system that addresses the above challenges with respect to realtime raster-based graphics synthesis. To summarize, this work makes the following contribution:

1. It describes a concept for the automated and view-dependent interpolation of global deformations based on the viewing angle of the user's virtual camera with respect to a reference plane.
2. It further presents an extension to global deformations that enables a user to define different geometric representations for different sections along a deformation curve and enables their image-based interpolation.

The remainder of this paper is structured as follows: Section 2 discusses related work. Section 3 introduces the concept of view-dependent global deformations. Section 4 describes steps to prepare the visualization. Section 5 outlines how to implement the concept as a realtime rendering technique. Section 6 consists of a performance evaluation, a preliminary user study, discusses problems and limitations, and presents ideas for future work. Section 7 concludes this paper.

2 RELATED WORK

Panoramic Imaging

Panoramic maps were introduced by H.C. Berann [3]. He combined handcrafted geographic with terrestrial depictions and different projection techniques to generate a new kind of map, which assists the user in the orientation task. This work was time-consuming and tedious. Premoze introduced a framework for the computer aided generation of panoramic maps [17]. It offers tools to assist the map-maker in the work flow of the hand-tailored maps. A semi-automatic approach to generate panoramic maps, which relies on global deformations, is presented in [24]. Falk et al. introduced a semi-automatic technique based on a force field that is extracted from the terrain surface [7]. Degener & Klein concentrate on parameters like occlusion and feature visibility in their automatic generation of panoramic maps [6]. All approaches combine non-linear perspectives in one final image, but rely on different techniques.

Non-linear Perspectives

Non-linear Perspectives can be achieved with different techniques: (1) Using non-standard, non-linear projection to produce a non-linear perspective image, or combine several images taken from different perspectives ([1], [23]). (2) Reflection on non planar surfaces and (3) Local or global space deformation [26]. The combination of different images to one final image as used in [1] and [23] can also be expressed by a space-deformation as introduced in [2]. The Single Camera Flexible Projection Framework of [4] is capable of combining linear, non-linear and handmade projections in realtime. The projections are described by a deformed viewing volume. Similar to free-form deformation (FFD [22]), the view frustum serves as lattice. Objects or viewing rays are deformed according to the deformation of the lattice. For the occlusion free visualization of driving routes Takahashi et al. rely on global space deformation [25].

On the one hand the mentioned techniques offer a broad and flexible definition of the projections, which enables the user to control nearly every facet of the final perspective. On the other hand a large number of non-intuitive parameters have to be controlled. Brosz et al. [2007] abstracts from these parameter by using a lattice. Similarly, we rely on a 2D B-Spline curve to control the 3D curve-based deformation.

Global Deformations

The work of Lorenz et al. [10] uses global deformation to generate non-linear perspectives. The geometry is mapped on two different planes, which are connected by a Bézier surface. The planes may vary in tilt, allowing for a combination of two different perspectives. Similar to panoramic maps, a mixture of cartographic maps and aerial images is used. The different stylization are seamlessly blended in the transition between the planes. Möser et al. [2008] extend this idea by using a more flexible Hermite curve to control the deformation. They also rely on a combination of aerial and cartographic images to apply a kind of generalization in the more distant parts of the scene.

Our approach is based on parametric curves, too. Instead of using a Hermite curve, we decided to use a B-Spline curve, because it offers more flexibility without the need of combining several curves. Furthermore, an arbitrary number of stylizations can be defined, which are not restricted to textures. Instead, we exploit the possibility of blending between different geometric representations generated by the generalization of 3D virtual city models as introduced in [8]. We introduce a view-dependent variation based on the work of Rademacher [19]. He defines key-deformation with associated key viewing points. Depending on the current viewpoint the key-deformations are interpolated. A similar approach is used by [5] for interactive stylized camera control. Another view-dependent variation of deformations is discussed in [12]. Here the global deformation is modified by a view or distant-dependent control function that can depend on a virtual camera.

3 VIEW-DEPENDENT GLOBAL DEFORMATIONS

Our approach consists of two main phases: (1) Rigging Visualization Presets: The user prepares discrete presets of the visualization. One visualization preset includes a deformation curve, the assignment of geometric representations to curve sections (tagging), and the definition of a viewing angle for which the preset is valid. (2) Realtime Visualization: During runtime the presets are interpolated using the camera parameters, which are manipulated during navigation or interaction with the 3D GeoVE.

3.1 Preliminaries

For our visualization we assume that a 3D GeoVE can be approximated by a 3D reference plane $R = (N, O) \in \mathbb{R}^3 \times \mathbb{R}^3$ defined by a normal vector N and a position vector O. Thus, and because of the isotropy of the global deformation variants used in this paper, a view setting for a virtual camera can be described by a viewing angle $\phi = cos(90 - C_D \cdot N)$ (Fig. 4).

To implement progressive or degressive perspectives [10] or hybrid forms [13], our approach uses B-Splines curves [20] instead of Hermite curves. In our experiments we use cubic B-Splines curves ($k = 4$) with four or six control points. In [9] it is argued that a smaller transition zone and linear segments would benefit the comprehension of such a visualization. This specific configuration is hard to implement using a single Hermite curve, but can be easily achieved using B-Splines curves with six control points, by setting two consecutive control points to the same position (Fig. 7).

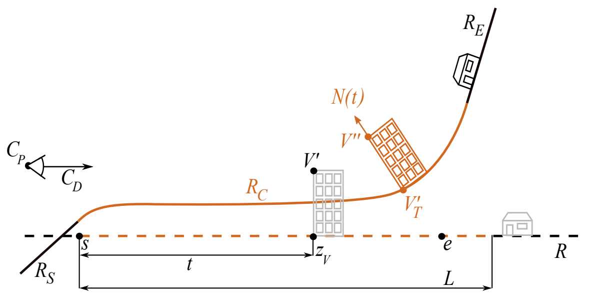

Figure 3: The reference plane R is separated by the parameter s and e into three sections: R_S, R_S and R_C. Based on the depth $z_{V'}$ of the vertex V along the camera direction C_D, the vertex is deformed onto one of the sections.

3.2 Application of Deformation Curves

We apply a global space deformation based on parametric curves, where the curve defines the deformation behavior. Therefore, R is subdivided into three sections (Fig. 3): (1) the curve-controlled section R_C, (2) a planar extension at the start R_S, and (3) a second planar extension at the end R_E. The deformation of R_C is controlled by a B-Spline curve $C(t)$ with a static open knot vector. Assuming that the control points B_i are fixed for a specific B-Spline, the position vector in curve-space $C(t) \in [0,1] \times [1,-1]$ only depends on the parameter t. To deform an input vertex $V = (x,y,z,w) \in \mathbb{R}^4$ we need to establish a mapping between V and t.

To establish the mapping, we first aligned V along the z-axis of the camera space $V' = V \cdot \mathbf{R}_A$. $\mathbf{R}_A$ rotates V around O by ϕ. After the rotation, every vertex is aligned along the viewing direction C_D of the virtual camera. The depth of V' is linearized between the user defined scalars for the start s and end e of the curve in camera space to compute $t \in [0,1]$. To account to the varying arc length L of the B-Spline curve in curve space, we perform a second normalization of t by L (Fig. 3). The rotation during the mapping is necessary, since otherwise a change of ϕ would lead to a different depth value of V and thus to a different mapping between V and t. Finally, the deformed vertex V'' is computed as follows:

$$V'' = \begin{cases} V' \cdot \mathbf{M}_S & t < 0 \\ V' \cdot \mathbf{M}_E & t > L \\ V' \cdot \mathbf{M}_{C(t)} & otherwise \end{cases} \qquad t = \frac{z_{V'} - s}{e - s} \cdot \frac{1}{L}$$

The deformation matrix $\mathbf{M}_{C(t)}$ consists of two separate translations $\mathbf{T}_{C(t)}$ and $\mathbf{D}_{C(t)}$, which are applied to V' sequentially. $\mathbf{T}_{C(t)}$ translates the vertex according to its position on the curve: Based on t a position vector $C(t)$ in curve space is computed. $C(t)$ is mapped back to camera space and used to translate V' onto R_C, yielding V'_T. Afterwards $\mathbf{D}_{C(t)}$ translates the vertex along the normal of the curve as follows: Based on the bi-normal B_x and the tangent $C'(t)$ the normal $N(t) = C'(t) \times B_x$ is computed. V'_T is translated along $N(t)$ by a distance d. Here, d denotes the distance of V' to its projection onto R. We just translate the position of the input vertex, because our deformation is a space deformation only. Operations which depends on vertex attributes, e.g. normals, are applied to the undeformed scene.

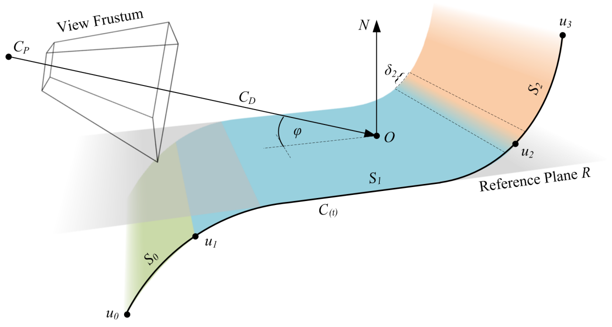

Figure 4: Exemplary parameterization of a deformation curve preset using four tag points (u_i).

To handle the cases of $t \notin [0,1]$ the deformation matrices $\mathbf{M}_S$ and $\mathbf{M}_E$ are applied accordingly to transform V' on R_S or R_E: If the extension plane is parallel to R the matrix is a translation matrix. Otherwise the matrix rotates V' on R_S or R_E. R_S is defined by the normal and position vector of the last B-Spline pont ($C(1)$) and R_E by the first point ($C(0)$).

Depending on the distribution of the control vertices and the knot vector of a B-Spline curve, a sampling with equidistant values t_1, t_2 and t_3 may not yield an equidistant distribution of points $P(t_1)$, $P(t_2)$ and $P(t_3)$, because a B-Spline curve is not arc-length preserving. This is distracting, since it will lead to a scaling error introduced by a straining or stretching of the geometric representation.

To guarantee a correct deformation behavior the curves must be re-parameterized. The approaches of [21] and [15] are not suited for our purposes because they either globally distribute the scaling error or are computational expensive. Instead, we decided to re-parameterize the parameter t similar to the method described in [14]. We sample the B-Spline curve in equidistant intervals and compute the arc-length of these segments. Based on the sampled length L and the according parameter t, the arc-length preserving parameter t' is computed by linear interpolation and stored in a lookup table.

3.3 Visualization Presets

Before we describe the tagging and interpolation of deformation curves, it is necessary to introduce the conceptual term *visualization preset*. As a preset we consider a single perspective (e.g., degressive or progressive). A preset P consists of the following components:

$$P = (C(t), \mathscr{T}, \mathscr{G}, \phi, \tau, s, e, a, b)$$

The set of all presets is denoted as $\mathscr{P}$, with $|\mathscr{P}| = m$. Besides a B-Spline curve $C(t)$ that is used to modify the global deformation, it contains an ordered list of tag points $\mathscr{T}$, a list of geometric representations $\mathscr{G}$ and the following scalar parameters (Fig. 4):

- ϕ: a camera angle, defined through the virtual camera and the reference plane R.
- τ: an angle interval around ϕ, where a preset is valid, i.e., no interpolation of the preset will occur.
- s, e: start and end of the deformation in eye-space. The interval is used to widen or narrow the curve-spaced deformation in camera-direction.
- $a, b \in [0,1]$: start and end of the geometry interpolation. This enables the user to define the geometry interpolation independent from the interpolation of the multi-perspective view.

3.4 Tagging of Deformation Curves

Our system enables the user to associate curve sections with different geometric representations. This can be useful for increasing or decreasing the visual complexity with respect to parts of the visualization. In [10], this was implied by blending between different type of textures within the transition zone and by omitting unimportant buildings. We extend this idea by blending between 3D geometry assigned to consecutive sections of a deformation curve (see Section 5.2). In our examples (Fig. 2 and 7) we use different levels of abstraction (LoA) automatically derived from the virtual city model of Berlin [8].

We can partition a deformation curve $C(t)$ into a number $l \geq 2$ of consecutive *styling sections* as part of the global set of sections $\mathscr{S}$:

$$S_i = (T_i, T_{i+1}, G), \qquad S_i \in \mathscr{S} \qquad G \in \mathscr{G}$$

Here, $i = 0, \ldots, l-1$ represents an index into the list of *tag-points* $\mathscr{T} = T_0, \ldots, T_l$ assigned to every preset P. The geometric representation for a section is denoted as G. A tag point T_i is further defined as follows:

$$T_i = (u, \delta) \quad u, \delta \in [0,1], \quad i = 0, \ldots, l \quad T_i \in \mathscr{T}$$

The position of the tag point on the curve is controlled via the parameter u. δ describes the length of the transition zone between two consecutive sections and is used for blending (see Section 5.2). We assume implicit fixed start and end tag points $T_0 = (0,0)$ at the curves start and $T_l = (1,0)$ at the curves end. Fig. 5 shows the different variants of a terrain model of the grand canyon and the associated active curve preset (inset).

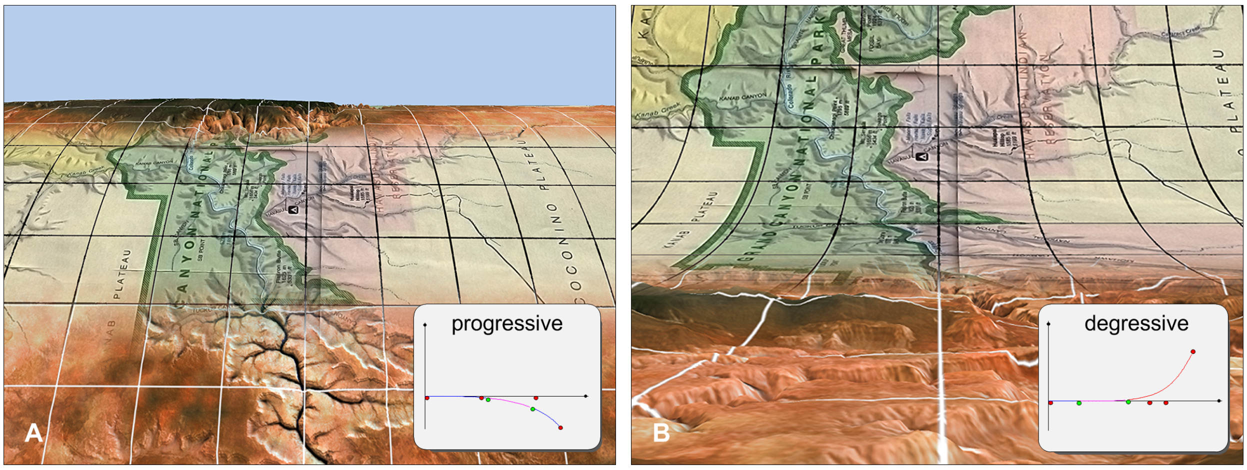

Figure 5: Styling section of a deformation curve with different models of the grand canyon. The inset shows the associated tag point and sections of the curve: The control points are depicted in red and the tag points are depicted green. The grid overlay was added to illustrate the deformation.

3.5 View-Dependent Curve Interpolation

The view-dependent curve interpolation, based on the camera angle ϕ, consists of two main steps: the *preset selection* and the *preset interpolation*. Given the viewing angle of the current virtual camera ϕ_a and the set of all presets $\mathscr{P}$, a selection function $s(\mathscr{P},\phi_a) = (P_S,P_T)$ delivers two presets as follows:

$$s(\mathscr{P},\phi_a) = (P_S,P_T) = \begin{cases} (P_i,P_{i+1}) & \phi_a \geq \phi_i \wedge \phi_a < \phi_{i+1} \\ (P_1,P_2) & \phi_a \leq \phi_1 \\ (P_{m-1},P_m) & \phi_a > \phi_m \end{cases}$$

for all $i = 1,\ldots,m$. This requires an ascending ordering of $\mathscr{P}$ by ϕ performed at the end of the rigging process. Given the viewing angle ϕ_a of the virtual camera and two presets P_S and P_T, the weighting factor σ is calculated as follows:

$$\sigma = clamp\left(\frac{\phi_a - \phi_S}{\phi_T - \phi_S}, 0, 1\right)$$

Given $\sigma \in [0,1]$, the source P_S and target preset P_T, the interpolation $P_I = p(P_S,P_T,\sigma)$ of the current preset P_I is performed by a linear interpolation of all control points: $B_{i,I} = B_{i,P_S} + \sigma \cdot (B_{i,P_T} - B_{i,P_S})$ as well as the respective tag points: $T_{i,I} = T_{i,P_S} + \sigma \cdot (T_{i,P_T} - T_{i,P_S})$.

Beside interpolating the curve related parameters, the geometric representations must also be interpolated. First the geometric representations of P_S and P_T are rendered into two texture-arrays, which are later blended according a factor $\beta \in [0,1]$, which is calculated as follows:

$$\beta = clamp\left(\frac{\sigma - a_{P_S}}{b_{P_S} - a_{P_S}}, 0, 1\right)$$

The interval $\left[a_{P_S}, b_{P_S}\right]$ defines in which section of the curve interpolation the geometric representations should be blended.

4 AUTHORING WORKFLOW

Our system supports interactive editing of the complete deformation curve parameterization and preset configuration at run time. To create a visualization, the user has to perform two steps: 1) adjust global settings required for every preset and 2) create or modify presets. According to Section 3.3 the user is required to select the number of control points and set the global number of tag points l, which are equally distributed over the length of the curve initially. This defines the number of styling sections implicitly.

After the global settings are defined, the user can modify the position and orientation of the virtual camera (ϕ) using standard interaction metaphors and edit the deformation curve parameters using direct manipulation of the curve control points. Further, the tag points can be moved along the deformation curve (which alters u) and the size of transition zone between two sections can be adjusted by altering δ. The user directly manipulates the tag points and the B-Spline control points using an interactive 2D widget (inset in Fig. 5). The scene models $\mathscr{G}$ can be loaded and assigned to the respective styling sections by dragging a geometric representation instance G to a respective styling section S. If the geometric representations of the different presets should not be interpolated over the complete interpolation interval, the user can adjust the parameters a and b. Finally, the start s and the end e parameters may be adjusted. These steps are then repeated for every preset.

Once all presets are prepared, the user can fine tune ϕ and τ to achieve the desired transitions. In terms of authoring effort, none of the depicted visualizations took more than three minutes to prepare. In all cases, the most time-consuming steps were the fine-tuning of the transition behavior and the modulation of the blending between the styling sections.

5 INTERACTIVE RENDERING

Our interactive visualization prototype is based on multi-pass rendering using OpenGL and OpenGL Shading language (GLSL). During multi-pass rendering, for each section the global space deformation is applied in the vertex shader. Each deformed geometric representation is written to an off-screen buffer, using Render-To-Texture (RTT) [16]. Finally the textures are composed. Details on the implementation are given in this section.

5.1 Global Deformation Computation

As described in Section 3.2 the deformation can be subdivided into two steps. First, every vertex V is aligned parallel to the camera viewing angle ϕ_a. To achieve this the viewing angle is recomputed on a per frame basis and the according rotation matrix $\mathbf{R}_A$ is passed to the vertex shader. Multiplying V with $\mathbf{R}_A$ yields V', which is projected on the reference plane R. Its initial distance d is stored in a shader variable. Second, the control point and tangent vector of the B-Spline curve is evaluated per vertex, to setup $\mathbf{M}_{C(t)}$. One possibility is to evaluate the B-Spline in the vertex shader. This implies, that the specific formulas to evaluate the parametric curves are known at compilation time and are fixed in the vertex shader code. A change of the parametric curve would lead to a change of the shader code. Instead, we decided to compute the position and tangent vector of the B-Spline curve off-line on the CPU. Thus, the B-Spline curve must be evaluated once a frame instead of once a vertex.

As mentioned in Section 3.2 the B-Spline must be arc-length parametrized. The lookup table is precomputed on the CPU and passed to the vertex shader, for the composition of styling sections, using a 32bit luminance texture. The texture lookup is performed by the parameter t, yielding the arc-length corrected values. The quality of the arc-length approximation depends on the number of precomputed samples. The bilinear interpolation during texture filtering provides a second parameter interpolation. This enables us to reduce the number of samples, without loosing precision. Experiments have shown that 2000 samples are sufficient for an arc-length preserving parametrization.

During the algorithm for arc-length parameterization we further compute the corrected position and tangent vectors of the B-Spline curve on the CPU. These values are stored in a texture that is later used as a lookup table in the vertex shader. The 2D-vectors $C(t)$ and $C'(t)$ are encoded in a 32-bit RGBA texture. The lookup table must be recomputed, if the setup of the parametric curve, e.g. the number or the position of the control points, changes. Thus, for a static curve setup, e.g. the user does not change the viewing angle of the virtual camera, no overhead is introduced. During view-dependent preset interpolation, the lookup table may be updated once per frame.

5.2 Compositing of Styling Sections

The composition consists of two steps: (1) Multipass RTT and (2) image-based composition in the fragment shader. To compose the potential different geometric representations of P_S and P_T, we choose an image-based compositing method, because it is generic and does not require knowledge of the underlying geometric representation. Every styling section of the presets is rendered into separate textures using RTT. Each texture contains RGBA information at viewport resolution. During rendering, a fragment shader adjust the α-value of a fragment according to the styling section boundaries defined by T_i and T_{i+1}, so that:

$$\alpha = \begin{cases} 1 & u_{T_i} + \delta_{T_i} \leq t \leq u_{T_{i+1}} - \delta_{T_{i+1}} \\ \frac{(u_{T_{i+1}} + \delta_{T_{i+1}}) - t}{2 \cdot \delta_{T_{i+1}}} & u_{T_{i+1}} - \delta_{T_{i+1}} < t \leq u_{T_{i+1}} + \delta_{T_{i+1}} \\ 0 & otherwise \end{cases}$$

After RTT is performed, the $2 \cdot (l-1)$ textures ($l-1$ textures per preset) are blended into the frame buffer. The blending of the layers is performed as follows: The first $(l-1)$ textures, encoding P_S, are blended based on α starting with the most distant styling section. The resulting fragment color is temporally stored. This procedure is repeated for the styling sections of P_E. Finally the two colors are blended based on β (see Section 3.5).

In addition thereto, Fig. 6 shows an application examples of the used stylization algorithms. In a preprocessing step, we compute light maps (ambient occlusion term only) for the complete model. At runtime, during the compositing step, we apply edge-detection based on normal and depth information of a fragment and we further unsharp-mask the depth buffer [11] to improve the perception of complex scenes by introducing additional depth cues.

6 RESULTS & DISCUSSION

6.1 Application Examples

We have tested our visualizations using different data sets. Besides photo realistic 3D city models, our ap-

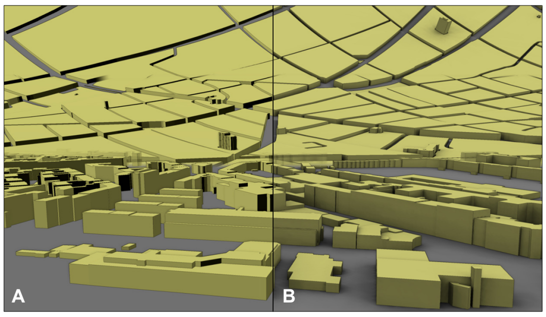

Figure 6: Comparison of applied stylization techniques for generalized virtual city models. A: Directional lighting and edge-enhancement. B: Precomputed ambient occlusion and edge-enhancement.

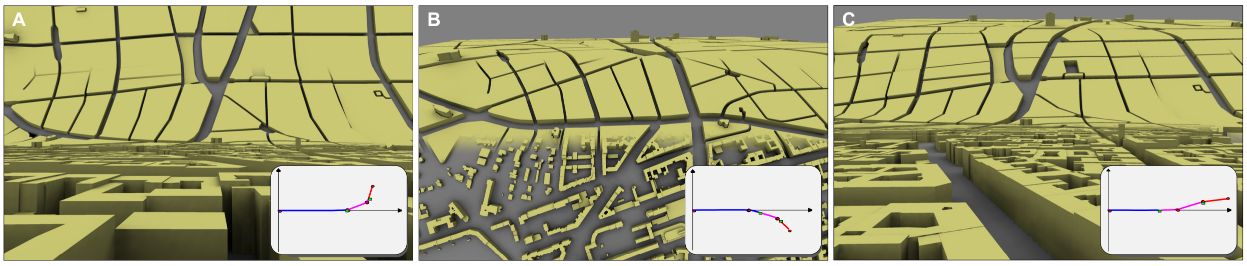

Figure 7: Exemplary visualization using B-Spline curves with six control points to enable hard transitions between three planar regions.

proach is in particular suitable for the depiction of different versions of generalized city models [8] (Fig. 2, Fig. 7). Despite the reduction of geometric complexity, the cell-based generalization also reduces the cognitive load of the user by displaying higher levels of abstraction. In comparison to the map-based stylization (Fig. 5), the generalized geometry is less expressive. The geometry must be enhanced, e.g., with labels, or textures, to communicate additional information to the user. Further, we use two model versions of the Grand Canyon with 524,288 triangles each. The first version uses a heightmap as well as an aerial image, while the second version represents a flat terrain with a tourist map applied. Fig. 5 shows the application of the model with a grid applied to emphasize the deformation.

During our experiments, we observed that the usage of more than three styling sections is rather distracting than informative to the user. A high number of sections also reduces the available space for each section. Thus, the amount of objects that can be visualized within a single section decreases. A similar effect arises if the transition zone between two sections (controlled by δ) is chosen to large. Further, the interval $[a_{P_S}, b_{P_S}]$, which control the blending of the geometric representations of P_S and P_T, should be set to initiate the blending briefly after the beginning or before the end of the curve interpolation.

To have a good control over the view-dependent behavior of the global deformation three visualization presets are sufficient, e.g., for a low, a medium and a high viewing angle. To gain more control or to fine tune the interpolation behavior we recommend to use more visualization presets.

6.2 Preliminary User Evaluation

We performed a preliminary user evaluation with 44 participants. The task is to navigate along a route with the help of a static image from a mobile navigation device. Therefore, we prepared 10 routes with a different complexity that partially contained landmarks. For each route we generated 4 visualizations using different perspectives: (1) orthographic (2D), (2) central (3D),(3) progressive and (4) degressive perspective. We presented the participants 26 image pairs. Each pair depicted the same route using two different perspectives. The user were asked which visualization they favor.

The results show that 80,7% of the participants favor the orthographic perspective instead of a central perspective. This is reasonable since a 2D map is a very established mean for navigation. Furthermore we observed that 76,1% prefer the degressive perspective instead of a central perspective. This indicates a demand for multi-perspective views for navigation. With our technique it becomes possible to combine the progressive perspective for a low viewing angle with the orthographic perspective for large viewing angles and thus provide the benefits of both visualization in one navigation tool.

6.3 Performance Evaluation

The performance tests are conducted using a NVIDIA GeForce GTX 285 GPU with 2048 MB video RAM on a Intel Xeon CPU with 2.33 GHz and 3 GB of main memory. The tests are performed at a viewport resolution of 1600 × 1200 pixels. Table 1 shows the results of our performance evaluation. All models are rendered using in-core rendering techniques with 8 × anti-aliasing. The performance mainly depends on the number of tag sections, thus the number of required rendering passes, and the geometrical complexity of the scenes attached to them. Due to the heavy usage of render-to-texture in the compositing steps, the performance also depends on the size of viewport. Here, the additional amount of graphics memory $O(l)$ required for a number of global styling section l can be estimated by: $O(l) = 2 \cdot l \cdot w \cdot h \cdot 4 \cdot p$ bytes. Our prototype uses a precision $p = 2$ byte per channel, which is sufficient for post-processing stylization.

Table 1: Comparative performance evaluation for different test scenes (in frames-per-second). The abbreviation LoA 0/1 names the configuration of a preset with two different models (LoA 0 and LoA 1) assigned to the two styling sections.

Preset config.	#Vertex	#Face	FPS
LoA 0/1	1,219,884	477,437	21
LoA 1/2	380,689	364,500	39
LoA 0/1/2	1,443,895	720,587	17

6.4 Limitations and Future Work

The presented approach implies a number of conceptual limitations. First, the number of control and tag points must be the same for each preset in a visualization. Further the visual quality of our approach relies on a sufficient vertex density of the geometric representations. We strive towards the application of hardware tessellation shader units to ensure this property for general scene geometry. Furthermore, the rendering concept is not optimized. At the moment each styling sections requires a single rendering pass. If two or more styling sections contain the same geometric representation, they can be treated as one single styling section reducing the number of rendering passes. The same applies for the geometry interpolation. The number of vertices can be further reduced by a culling algorithm based on the boundaries of the styling sections.

7 CONCLUSIONS

This paper presents a concept and interactive rendering technique for view-dependent global deformations that can be used for the effective visualization of 3D geovirtual environments, such as virtual 3D city and landscape models. It presents an approach for a view-dependent parameterization and interpolation of global deformations based on B-Spline curves. The application of such parametrized curves offers the possibility to customize or extend traditional perspectives, e.g. degressive or progressive perspectives, in a comprehensible and flexible way. Further, the definition of camera-dependent presets and their automatic interpolation overcomes the restriction of existing multi-perspective visualization. In addition, we provide a concept for assigning different geometric representations to specific sections of a curve, which offers more freedom of design. We further present a prototypical implementation that enables hardware-accelerated realtime image synthesis as discussed in our performance evaluation.

ACKNOWLEDGMENTS

This work has been funded by the German Federal Ministry of Education and Research (BMBF) as part of the InnoProfile research group "3D Geoinformation". The authors like to thank Tassilo Glander for providing the data sets of the generalized city model of Berlin and Haik Lorenz for his support and critical comments.

REFERENCES

[1] Maneesh Agrawala, Denis Zorin, and Tamara Munzner. Artistic multiprojection rendering. In *Proc. of the EG Workshop on Rendering Techniques*, pages 125–136, 2000.

[2] Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84*, pages 21–30, New York, NY, USA, 1984. ACM.

[3] Heinrich Caesar Berann. The world of h.c. berann. web site.

[4] John Brosz, Faramarz F. Samavati, M. T. Carpendale Sheelagh, and Mario Costa Sousa. Single camera flexible projection. In *NPAR '07*, pages 33–42, New York, NY, USA, 2007. ACM.

[5] Nicholas Burtnyk, Azam Khan, George Fitzmaurice, Ravin Balakrishnan, and Gordon Kurtenbach. Stylecam: interactive stylized 3d navigation using integrated spatial & temporal controls. In *UIST '02*, pages 101–110, New York, NY, USA, 2002. ACM.

[6] Patrick Degener and Reinhard Klein. A variational approach for automatic generation of panoramic maps. *ACM Trans. Graph.*, 28(1):1–14, 2009.

[7] Martin Falk, Tobias Schafhitzel, Daniel Weiskopf, and Thomas Ertl. Panorama maps with non-linear ray tracing. In *GRAPHITE '07*, pages 9–16, New York, NY, USA, 2007. ACM.

[8] Tassilo Glander and Jürgen Döllner. Abstract representations for interactive visualization of virtual 3d city models. *Computers, Environment and Urban Systems*, 33(5):375 – 387, 2009.

[9] Markus Jobst and Jürgen Döllner. Better perception of 3d-spatial relations by viewport variations. In *VISUAL '08*, pages 7–18, Berlin, Heidelberg, 2008. Springer-Verlag.

[10] Haik Lorenz, Matthias Trapp, Jürgen Döllner, and Markus Jobst. Interactive multi-perspective views of virtual 3d landscape and city models. In *AGILE Conf.*, pages 301–321, 2008.

[11] Thomas Luft, Carsten Colditz, and Oliver Deussen. Image enhancement by unsharp masking the depth buffer. *ACM Trans. Graph.*, 25(3):1206–1213, 2006.

[12] D. Martín, S. García, and J. C. Torres. Observer dependent deformations in illustration. In *NPAR '00*, pages 75–82, New York, NY, USA, 2000. ACM.

[13] Sebastian Möser, Patrick Degener, Roland Wahl, and Reinhard Klein. Context aware terrain visualization for wayfinding and navigation. *Computer Graphics Forum*, 27(7):1853–1860, 2008.

[14] Qunsheng Peng, Xiaogang Jin, and Jieqing Feng. Arc-length-based axial deformation and length preserved animation. In *CA '97*, page 86, Washington, DC, USA, 1997.

[15] John W. Peterson. Abstract arc length parameterization of spline curves.

[16] Matt Pharr and Randima Fernando. *GPU Gems 2*. Addison-Wesley Professional, 2005.

[17] Simon Premoze. Computer generated panorama maps. In *ICA Mountain Cartography Workshop*, 2002.

[18] Huamin Qu, Haomian Wang, Weiwei Cui, Yingcai Wu, and Ming-Yuen Chan. Focus+context route zooming and information overlay in 3d urban environments. *IEEE TVCG*, 15(6):1547–1554, 2009.

[19] Paul Rademacher. View-dependent geometry. In *SIGGRAPH '99*, pages 439–446, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[20] Richard Franklin Riesenfeld. *Applications of b-spline approximation to geometric problems of computer-aided design.* PhD thesis, Syracuse, NY, USA, 1973.

[21] David F. Rogers. *An Introduction to NURBS: With Historical Perspective*. Morgan Kaufmann, 2000.

[22] Thomas W. Sederberg and Scott R. Parry. Free-form deformation of solid geometric models. *SIGGRAPH*, 20(4):151–160, 1986.

[23] Karan Singh. A fresh perspective. In *Proc. Graphics Interface*, pages 17–24, May 2002.

[24] Shigeo Takahashi, Naoya Ohta, Hiroko Nakamura, Yuriko Takeshima, and Issei Fujishiro. Modeling surperspective projection of landscapes for geographical guidemap generation. *Computer Graphics Forum*, 21:2002, 2002.

[25] Shigeo Takahashi, Kenichi Yoshida, Kenji Shimada, and Tomoyuki Nishita. Occlusion-free animation of driving routes for car navigation systems. *IEEE TVCG*, 12:1141–1148, 2006.

[26] Scott Vallance and Paul Calder. Multi-perspective images for visualisation. In *VIP '01*, pages 69–76, Darlinghurst, Australia, Australia, 2001. Australian Computer Society, Inc.