Real-Time Rendering of Water Surfaces with Cartography-Oriented Design

Amir Semmo[1] Jan Eric Kyprianidis[2] Matthias Trapp[1] Jürgen Döllner[1]

[1]Hasso-Plattner-Institut, Germany* [2]TU Berlin, Germany*

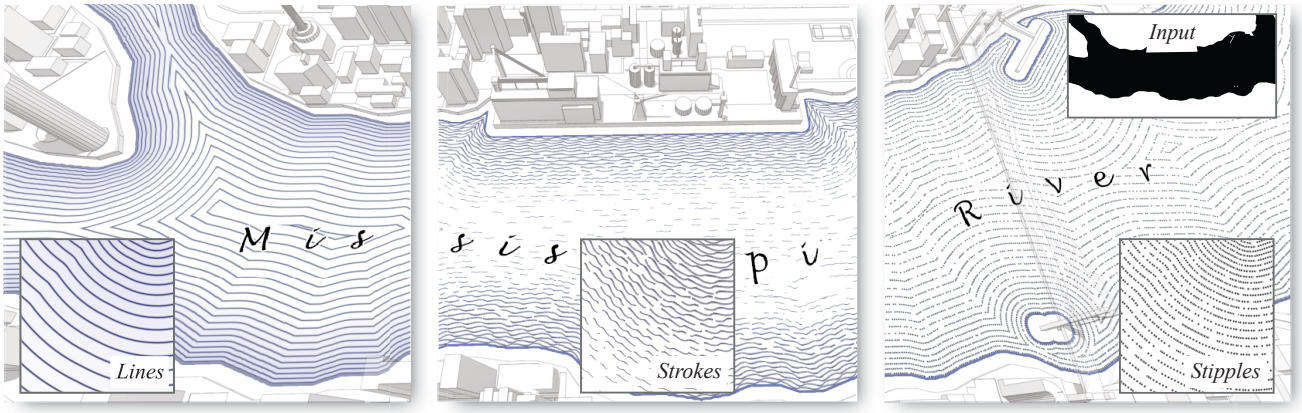


Figure 1: *Illustrative rendering techniques implemented in our system: waterlining, contour-hatching, water stippling, and labeling.*

Abstract

More than 70% of the Earth's surface is covered by oceans, seas, and lakes, making water surfaces one of the primary elements in geospatial visualization. Traditional approaches in computer graphics simulate and animate water surfaces in the most realistic ways. However, to improve orientation, navigation, and analysis tasks within 3D virtual environments, these surfaces need to be carefully designed to enhance shape perception and land-water distinction. We present an interactive system that renders water surfaces with cartography-oriented design using the conventions of mapmakers. Our approach is based on the observation that hand-drawn maps utilize and align texture features to shorelines with non-linear distance to improve figure-ground perception and express motion. To obtain local orientation and principal curvature directions, first, our system computes distance and feature-aligned distance maps. Given these maps, waterlining, water stippling, contour-hatching, and labeling are applied in real-time with spatial and temporal coherence. The presented methods can be useful for map exploration, landscaping, urban planning, and disaster management, which is demonstrated by various real-world virtual 3D city and landscape models.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Display Algorithms

Keywords: water surfaces, illustrative rendering, cartography-oriented design, contour-hatching, waterlining, water stippling

1 Introduction

Water surfaces represent key elements in mapmaking and surveying because they shape our world and convey important information to a number of domains, including hydrology, urban planning and environmental sciences. Hand-drawn illustrations of such surfaces are often carefully designed to help a viewer explore the geospatial environment. Usually, these designs effectively establish land-water distinction to facilitate orientation, navigation, or analysis tasks.

Yet the many different shapes of water surfaces pose a number of challenges that have required contemporary craftsmanship and design skills. Typical challenges include *(1) the symbolization of the land-water interface* using *(2) design elements (e.g., strokes) that exactly align with the shorelines* to effectively provide figure-ground and express motion. Over centuries, cartographers have developed illustration techniques and design principles that address these challenges [Imhof 1972; Robinson et al. 1995; Merian 2005]. Certain techniques have become abundantly used in modern cartography, such as waterlining, hatching, or water stippling; and most of them express aesthetical appeal, provide excellent figure-ground balance, and establish a sense of motion [Christensen 2008; Huffman 2010]. For instance, fine solid lines are placed parallel to shorelines to effectively communicate shoreline distances (Figure 1 left).

To date, computer-generated illustrations of water surfaces are mainly based on photorealistic rendering techniques [Darles et al. 2011]. Approaches in illustrative rendering have been subject to cartoon-like water effects [Eden et al. 2007; Yu et al. 2007] or other primary cartographic elements, such as terrain [Bratkova et al. 2009; Buchin et al. 2004], vegetation [Deussen and Strothotte 2000; Coconu et al. 2006], or buildings [Döllner and Walther 2003], but have neglected the many challenges water exhibits for map design.

This paper presents an interactive system for rendering water surfaces with cartography-oriented design that addresses the aforementioned challenges. For this, our work makes the following contributions. First, we identify and describe design principles from traditional and modern cartography to create more effective illustrations of water surfaces. Based on these findings, Euclidean distance maps are computed using a novel feature-aligned distance transform to derive principal curvature directions of complex water shapes. Using this information, we contribute real-time rendering techniques for waterlining, contour-hatching, water stippling and labeling (Figure 1) that facilitate a view-dependent level-of-abstraction [Semmo et al. 2012]. Finally, we tested these rendering techniques with various

*http://www.hpi3d.de | http://www.cg.tu-berlin.de

CAe 2013, July 19 – 21, 2013, Anaheim, California.

real-world virtual 3D city and landscape models. Our results reveal potential applications within 3D geovirtual environments for map exploration, landscaping, urban planning, and flooding simulation.

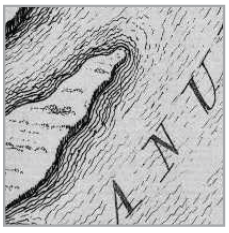
The remainder of this paper is structured as follows. Section 2 summarizes design principles derived from hand-drawn maps and textbooks on map design. Section 3 reviews related works on texture synthesis and illustrative rendering. Section 4 presents our methods used for rendering water surfaces according to the identified design principles, whose implementation details and results are presented in Section 5. Finally, Section 6 concludes this paper.

2 Design Principles from Cartography

Well-designed illustrations of water surfaces provide figure-ground, establish a sense of motion, and communicate meta-information (e.g., water names). A variety of illustration techniques have been developed by cartographers whose design principles address feature-aligned texturing and symbolization. However, most cartographers develop and vary their own illustration styles. Therefore, we analyzed the work by famous cartographers like Matthäus Merian and Jouvin de Rochefort printed in map collections (e.g., [Merian 2005]), and textbooks on map design and thematic cartography [French 1918; Imhof 1975; MacEachren 1995; Kraak and Ormeling 2003; Tyner 2010]. From this analysis, we extracted the following design principles, which we used for our illustrative rendering techniques:

Waterlining. Waterlining became popular in the first half of the 20th century for lithographed maps, because areas of solid color tones could not be produced at that time. With this technique, *(P1) fine solid lines are drawn parallel to shorelines, and the spacing between succeeding lines gradually increase* [French 1918]. A common mistake is to "make the lines excessively wavy or rippled" [French 1918] or the distance between lines with insufficient continuity. If drawn with care, waterlining provides dynamism and effectively propagates distance information [Christensen 2008; Huffman 2010].

Water Stippling. Another conventional technique for hand-drawn black-and-white maps is water stippling [Tyner 2010]. Similar to waterlining, distance information is propagated by aligning small dots with non-linear distances to shorelines. Compared to stippling in traditional artwork, water surfaces are depicted by *(P2) stipples with varying density that irregularly overlap along streamlines* to establish a sense of motion. In perspective views, some cartographers draw *(P3) stipples with higher density at occluded areas (e.g., near bridges) to improve depth perception, or with varying density to symbolize flow velocity*.



Contour-Hatching and Vignetting. Contour-hatching has been widely used to balance the accurate propagation of distance information and establishment of motion: by using *(P4) individual strokes that are placed with high density near shorelines and complemented by loose lines placed with increasing irregularity towards the middle stream*. In contrast to waterlines, *(P5) excessively wavy strokes are drawn to express motion*. Alternative illustrations use non-feature-aligned cross-hatches for land-water distinction. These methods have been replaced in the second half of the 20th century by color tones and drop shadows [Tyner 2010]. Today, coastal vignettes with solid color tones are used to establish figure-ground but, in contrast to contour-hatching, fail to express water movements.

Labeling. Labels are design elements used in cartography to enrich maps with meta-information. By convention, *(P6) cartographers depict names of water features with italic (slanted) letters* to distinguish them from land features for which upright letters are used [Tyner 2010]. Typically, *(P7) names follow principal curvature directions and are placed within water surfaces* [Imhof 1975] to ensure legibility.

Symbolization. Symbolization is a common practice in cartography to reflect data and phenomena [Imhof 1972]. To date, standardized symbolization for water surfaces has not been established. However, certain conventions have been used over the years, including *(P8) the irregular placement of signatures with area-wide coverage* to communicate water features (e.g., wetland, saltwater vs. freshwater), or the placement of glyphs along streamlines to symbolize the flow direction of rivers.

For map design, blue is established as a conventional color tone, and our work considers the identified design principles.

3 Related Work

Our work is related to previous works on texture synthesis, illustrative rendering, and cartography-oriented design.

Feature-guided Texture Synthesis. From our analysis in Section 2, we observed that texture features are aligned with shorelines to symbolize the land-water interface. Geometric properties of complex shapes can be reconstructed quite effectively using distance fields [Frisken et al. 2000]. Our work uses distance maps to synthesize waterlines and align stipples with shorelines in real-time. Most algorithms use vector propagation to compute these maps by an approximate Euclidean distance transform [Danielsson 1980] (e.g., jump-flooding [Rong and Tan 2006]). We use the fast, workload efficient *Parallel Banding Algorithm* (PBA) to compute exact distance maps on the GPU [Cao et al. 2010].

Feature-guided texturing based on principal curvature directions can significantly improve shape recognition [Girshick et al. 2000]. For an overview on this topic we refer to the survey by *Wei et al.* [2009]. Recent approaches use normal-ray differential geometry [Kim et al. 2008b] or diffusion techniques [Xu et al. 2009] to derive principal curvature directions, or use learning-based approaches [Kalogerakis et al. 2012; Gerl and Isenberg 2013] to align textures to salient feature curves. However, these methods either require significant (pre-)processing time or do not provide spatial and temporal coherence. By contrast, we present the concept of a *feature-aligned distance transform* that provides continuous Euclidean distance values in a tangential direction to the shorelines. We use a GPU-based flooding algorithm to compute a feature-aligned distance map in real-time. Together with bilinear texture interpolation [Green 2007], this map can be used to parameterize and place texture features along shorelines with frame-to-frame coherence.

Non-Photorealistic Rendering. Coastal vignettes and waterlines are used for cartographic line generalization [Christensen 1999] and in geoinformation systems to improve figure-ground perception.

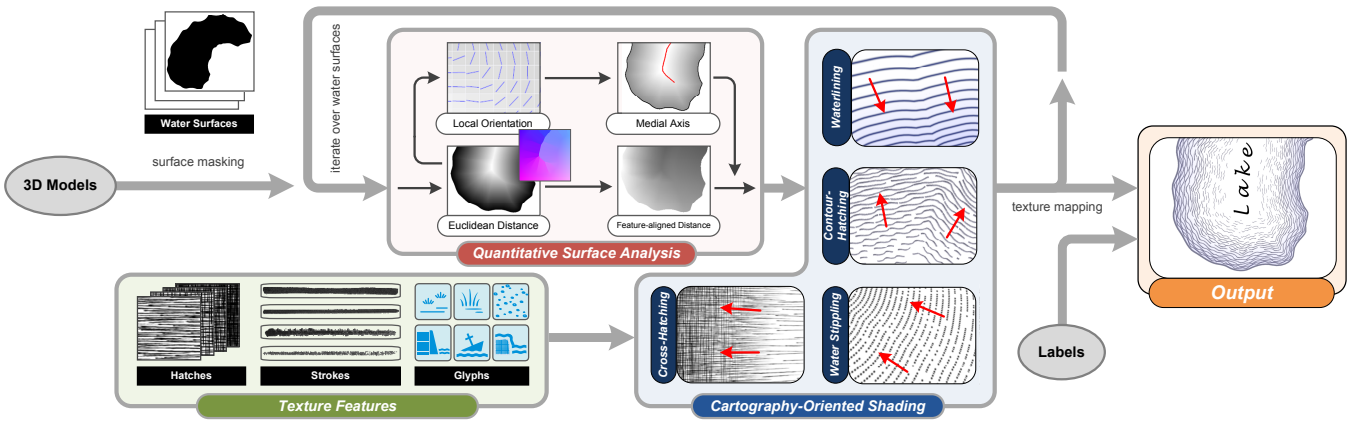


Figure 2: *Schematic overview of our system, which implements cartography-oriented shading using the results of quantitative surface analysis.*

Stippling is a well-studied field in illustrative rendering for digital half-toning. Conventional approaches represent local tone by a well-spaced placement of small dots [Kyprianidis et al. 2012]. Previous work proposed feature-guided image stippling [Kim et al. 2008a; Kim et al. 2010] that is adapted to the gradient direction of distance maps. Water stippling works similarly, but does not match density distributions to local tones (e.g., by blue noise) because dots irregularly overlap with varying density (P2). For this, we propose an enhancement to Glanville's [2004] texture bombing algorithm that aligns water stipples with waterlines and renders them in real-time.

Feature-guided hatching has received significant attention in previous works and comprises user-defined [Salisbury et al. 1997], patch-based [Praun et al. 2000; Webb et al. 2002], shading-based [Praun et al. 2001], or learning-based [Kalogerakis et al. 2012; Gerl and Isenberg 2013] algorithms. The main difference to our approach is that texture coordinates are obtained by Euclidean and feature-aligned distance maps, which gives us more artistic control over parameterizing individual strokes within real-time shaders. For cross-hatching, tonal art maps [Praun et al. 2001] are aligned to Euclidean distance maps according to the view distance. This way, a continuous level-of-abstraction [Semmo et al. 2012] can be achieved that significantly reduces visual clutter at high view distances. A level-of-abstraction may also be combined with shape simplifications to produce aesthetic renditions of digital maps [Isenberg 2013].

Cartography-Oriented Design. Illustrative rendering of water surfaces is not a new approach. Previous works used colorization, edge enhancement, and texturing for cartoon-like water effects (e.g., ripples) and to emphasize liquid movement [Eden et al. 2007; Yu et al. 2007]. However, these rendering styles do not relate to traditional map design in terms of figure-ground organization for map exploration, navigation, or landscaping. Rendering with cartography-oriented design has been subject to other primary map elements, such as terrain [Bratkova et al. 2009], trees [Deussen and Strothotte 2000], and buildings [Döllner and Walther 2003], as well as landscape [Coconu et al. 2006] and city models [Jobst and Döllner 2008]. We complement these techniques by our work on water surfaces and exemplify how cartography-oriented visualization of 3D geovirtual environments can be achieved by combining our results with traditional relief presentations of landscapes [Buchin et al. 2004].

Internal labeling has been addressed in the visualization of virtual 3D scenes. Previous works compute geometric hulls [Maass and Döllner 2008] or derive medial axes based on a distance transform [Götzelmann et al. 2005; Ropinski et al. 2007; Cipriano and Gleicher 2008] for shape-aligned labeling. Our work also uses distance maps to align font glyphs with the shoreline distance and orientation.

4 Method

An overview of our system is shown in Figure 2. The input data consists of a set of 2D or 3D water surfaces that are typically defined as triangular irregular networks. Using orthographic projections, the models' shapes were captured in 2D binary masks to facilitate quantitative surface analysis that works in image-space. This analysis includes the computation of Euclidean and feature-aligned distance maps by iteratively propagating distance information in the normal and tangent directions of shorelines. This information was used for cartography-oriented shading, including waterlining, contour-hatching, water stippling, and labeling. To enable a continuous level-of-abstraction [Semmo et al. 2012], the shading results were parameterized, blended, and mapped onto the surfaces according to the view distance. Because water surfaces are processed separately, our system can be seamlessly embedded into existing rendering systems, or combined with rendering techniques that pre-process the 3D virtual environment (e.g., terrain [Buchin et al. 2004]).

4.1 Quantitative Surface Analysis

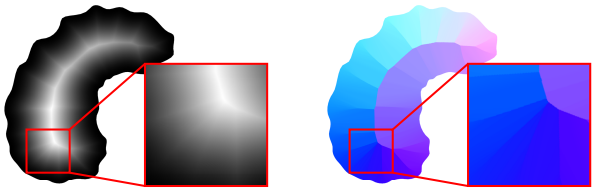
Our goal is to provide quality, interactive illustrations of water surfaces that comply with the identified design principles (Section 2). This section provides background on how geometric properties of water surfaces can be derived using distance transforms. It includes a novel feature-aligned distance transform that is used to align individual strokes with the orientation of shorelines.

4.1.1 Euclidean Distance Transform

Euclidean distance information was computed to determine parts of a water surface with equal shoreline distance. Let $I : \mathbb{R}^2 \rightarrow \{0, 1\}$ denote a water surface captured as a binary image, with $I(p) = 0$ marking water areas, and $I(p) = 1$ marking land areas by pixels $p \in I_D$ (see Figure 2 top left). A distance transform of I defined as:

$$\omega_I(p) = \min_{q \in I_D} \left(||p - q|| + \chi(q) \right) \text{ with } \chi(q) = \begin{cases} 0 & \text{if } I(q) = 1 \\ \infty & \text{otherwise} \end{cases}$$

obtains the minimum Euclidean distance of each pixel to a shoreline. A fast, parallel implementation to compute this information as a *distance map* D (Figure 3a) is based on the PBA [Cao et al. 2010]. Iteratively propagating distance information with the PBA was also performed to obtain the nearest shoreline position as directional information (Figure 3b). Subsequently, $D(p)$ was used as a lookup function for shoreline distances $d \in \mathbb{R}^+$, and $D_b(p)$ to lookup the nearest shoreline position $b \in I_D$.

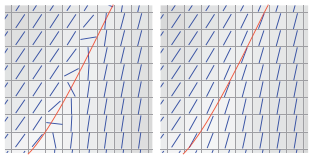


(a) *shoreline distance* **(b)** *shoreline direction*

Figure 3: *Exemplary visualization of normalized shoreline distances and shoreline directions for a given water surface.*

4.1.2 Local Orientation Estimation

The estimation of local orientation is based on the image gradients of the distance map D. A popular choice to approximate the directional derivatives in x- and y-direction is the Sobel filter which, however, yields non-smooth tangent information on the medial axes because opposite gradients cancel out (Figure 4 left). A simple alternative is to use the smoothed structure tensor [Brox et al. 2006] and perform an eigenanalysis to obtain gradient and tangent information. This leads to more stable estimates of the local orientation (Figure 4 right).

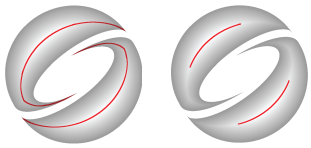


Sobel Filter *Structure Tensor*

Figure 4: *Tangential field.*

4.1.3 Medial Axes Computation

The medial axes were derived from the distance map D [Cao et al. 2010] to align design elements (e.g., labels) along the middle stream of water surfaces. Basically, the medial axes were obtained by comparing and thresholding the directions to the nearest shorelines in the local neighborhood for each $p \in I_D$. For this, the unsigned gradient orientation $n \in \mathbb{R}^2$ of the smoothed structure tensor is used:



θ=0.75π δ=0.0 θ=0.75π δ=0.8

Figure 5: *Medial axes.*

$$b^+ = ||p - D_b(p+n)||\,, \quad b^- = ||p - D_b(p-n)||\,.$$

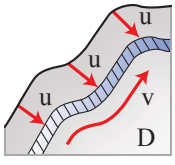
Thresholding the angle between b^+ and b^- then yields an approximate of the medial axes:

$$\arccos\left(b^+ \cdot b^-\right) > \theta \in [0, \pi].$$

In addition, the shoreline distance was thresholded by $\delta \in \mathbb{R}^+$ to avoid placing design elements too close to shorelines. For all the examples in this paper, we use $\theta = 0.75\pi$ and $\delta = 0.8$ (Figure 5).

4.1.4 Feature-aligned Distance Transform

Contour-hatching for water surfaces is a complex problem, since the properties of individual strokes (e.g., length, spacing) vary with the shoreline distance (P4). A typical approach is to define orientation fields on a surface to guide an example-based texture synthesis to salient feature curves [Wei et al. 2009]. Yet animating individual strokes on water surfaces requires fine control over the parameterization and placement per rendering pass, in particular to simulate water movements. Our approach parameterizes the level-set curves of distance map D to obtain Euclidean distance values along its tangential field. By parameterizing these



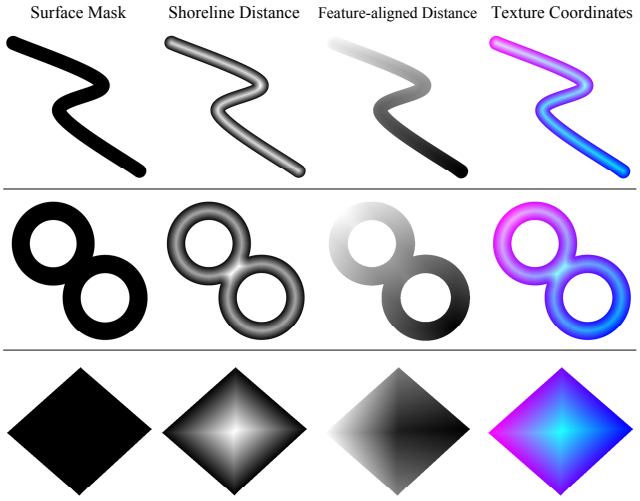
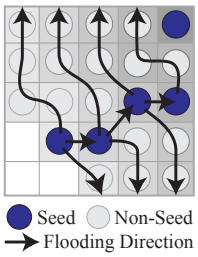


Figure 6: *Exemplary distance maps computed by our system.*

feature-aligned distances (v-coordinate) and the shoreline distances (u-coordinate), they are directly used as texture coordinates within real-time shaders (Figure 6).

Algorithm. For computing a *feature-aligned distance map* T, we use an approach similar to vector propagation [Danielsson 1980]. Using the non-normalized distance map D, level sets correspond to the integral part of the shoreline distances (e.g., $\lfloor d \rfloor = 0$ for the zero level sets). Starting with the shorelines, random pixels are selected as seed points from which (1) Euclidean distances are propagated along the level sets, and (2) seed point information is propagated to the inner level sets. These two steps are repeated for each level set until no more pixels are available for processing. We implemented a parallel algorithm by iteratively flooding distance information within the local neighborhood (e.g., 3×3) of a pixel, which dynamically propagates seed points during distance map construction (Figure 7). An efficient parallel algorithm for normalization of this map is based on a reduction [Nehab et al. 2011].



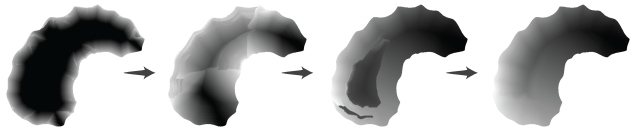
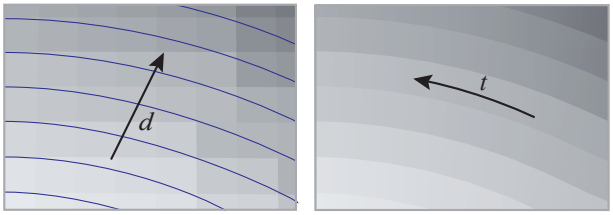


Figure 7: *Intermediate results of our algorithm that iteratively computes a feature-aligned distance map on a GPU (*256 $\times$ 256 *pixels,* 503 *iterations in total).*

Discussion. Exemplary results show that our approach provides continuous feature-aligned distance values (Figure 6/7). Contrary to texture synthesis based on energy minimization (e.g., [Xu et al. 2009]), ours does not provide continuity across the level sets of a surface. But since individual texture features (e.g., hatches) are aligned with the level-set curves, no such constraint is required. This allows us to perform an exact distance transform since no compression or stretching is required to meet continuity in all major directions of a surface. Because texture features can only be placed on the level sets, choosing an adequate distance map resolution is important to balance rendering quality and performance. On the one

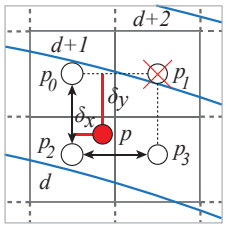


(a) *discrete input + distance lines* **(b)** *continuous output*

Figure 8: *Bilinear filtering of a feature-aligned distance map.*

hand, vector propagation along the level sets performs non-linearly with the map resolution, and optimization techniques known from jump flooding [Rong and Tan 2006] are less helpful since highly curved sections require small step widths. On the other hand, too low map resolutions insufficiently approximate the shorelines' directions. As a compromise, we utilized bilinear sampling for a piecewise-linear approximation of shoreline and feature-aligned distances. This approach has been proven effective for the magnification of glyph contours, even with low-resolution distance maps [Green 2007]. Because bilinear sampling is able to accurately reconstruct distance information, feature-aligned distance maps of up to 256×256 pixels can be used for rendering and computed in real-time using our GPU-based flooding algorithm (Section 5).

Bilinear Sampling. In contrast to signed distance maps, the accurate reconstruction of feature-aligned distance values requires a modified version of bilinear sampling to avoid filtering across the level sets of D. For this, the shoreline distance d for a point $p \in I_D$ is determined and compared to the distance information of the four samples p_0 to p_3:



$$T(p) = A + B - ((1-\delta_y)Bg(A) + \delta_y Ag(B))$$
$$A = (1-u_1)p_0 + u_1 p_1 \quad u_1 = (g(p_0)(\delta_x - 1) + 1)g(p_1)$$
$$B = (1-u_2)p_2 + u_2 p_3 \quad u_2 = (g(p_2)(\delta_x - 1) + 1)g(p_3)$$

where $g(q) = 0$ with $q \in \{p_0, p_1, p_2, p_3\}$ if p, q correspond to different level sets to omit a pixel from interpolation, otherwise $g(q) = 1$. As is shown in Figure 8, this modified version provides continuous feature-aligned distance values on the level sets.

4.2 Shading Techniques

The results of the quantitative surface analysis are utilized for cartography-oriented shading (Section 2). The following techniques utilize pixel shaders and texturing combined with bilinear sampling to accurately reconstruct distance information. These techniques can be parameterized in terms of tone and density to provide a view-dependent level-of-abstraction (see the Appendix).

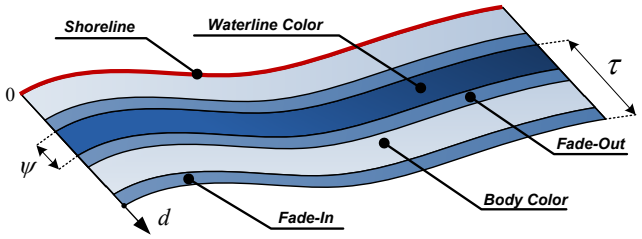


Figure 10: *Waterlining parameterized by shoreline distance.*

4.2.1 Waterlining and Water Stippling

In the following, a non-normalized version of distance map D is used to independently apply waterlining and stippling from a water surface's scale (e.g., oceans vs. lakes). To comply with non-equidistant interspaces (P1), target distance values $\varphi(d)$ are computed using a non-linear step function:

$$\varphi(d) = \frac{(\lfloor (s \cdot d)^e + h \rfloor - h)^{1.0/e}}{s}.$$

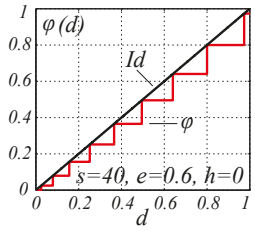


Figure 9: *Non-linear step function $\varphi(d)$.*

The spacing of φ can be parameterized by $e, s \in \mathbb{R}^+$ to define a corresponding number of steps in the interval of D (Figure 9). In addition, these steps can be shifted along d using $h \in [0, 1]$.

Waterlining. Waterlines correspond to shaded areas of a water surface with equal shoreline distance. To render waterlines, the distances $\tau \in \mathbb{R}^+$ between the positions obtained by $\varphi(d)$ are thresholded by a corresponding width $\psi \in \mathbb{R}^+$, and padded by fade-in and fade-out intervals (Figure 10) for antialiasing, and to provide smooth transitions. For a continuous level-of-abstraction, $\varphi(d)$ and ψ are parameterized by the view distance. At high view distances, this significantly reduces the number of rendered waterlines and provides a smooth transition while zooming in (see the accompanying video).

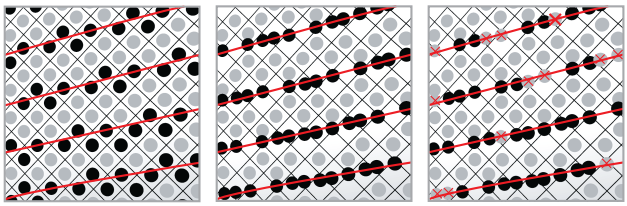
Water Stippling. Water stippling refers to placing small dots with irregular distribution along waterlines to convey shape and motion. Our algorithm uses an enhanced variant of Glanville's [2004] texture bombing to place water stipples with feature-aligned distribution and irregular density. The basic idea of texture bombing is to randomly place glyphs in regularly distributed grid cells. We extend this algorithm by three phases; *stipple selection*, *stipple displacement*, and *stipple filtering* (Figure 11). Instead of using a random placement of stipples, offsets are computed that align them with the waterlines of D. Our algorithm starts with stipples that are centered in regularly distributed grid cells and mapped onto a water surface (Figure 11a).

1. Stipple Selection: Stipples within grid cells that cross a waterline were selected for further processing (Figure 11a). For this, the distance to the next waterline ($d - \varphi(d)$) is thresholded.

2. Stipple Displacement: The gradient direction of D was used to compute the approximate target position to the nearest waterline. If origin and target positions correspond to the same grid cell (first phase), a stipple was displaced towards the target position. This results in stipples lined up with the waterlines (Figure 11b).

3. Stipple Filtering: The displacement of stipples in the gradient direction of D increased the irregular distribution along waterlines. To render stipples with non-regular intervals, noise or pseudo-random numbers were used for additional filtering (Figure 11c). Figure 12 exemplifies that this improves randomness.

Up to this point, stipples might be rendered with low density because waterlines force them to split into multiple directions. To regularize the density near coastal areas, the phases 1-3 were repeated to place additional layers of stipples within slightly shifted grid cells. We found that two layers were sufficient to meet this requirement (Figure 12c). We experimented with different parameterizations to locally vary the stipple density and tone, for example using tonal art maps to symbolize flow velocity (P3). In addition, iterative application of our algorithm with shifted step functions can be used to indicate highlights or shadowed areas. For instance, a second pass with $\varphi(d)$ shifted halfway by $h = 0.5$ was used to place additional stipples at occluded areas (e.g., near bridges, Figure 19) to improve depth perception of a virtual 3D scene (P3).



(a) *selection* **(b)** *displacement* **(c)** *filtering*

Figure 11: *Schematic overview of the water stippling phases. Waterlines are marked as red lines, rendered stipples as black dots.*

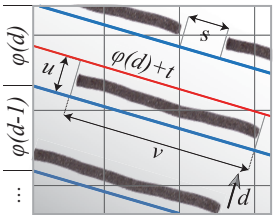
(a) *overview* **(b)** *middle stream* **(c)** *coastal area*

Figure 12: *A result of our water stippling technique showing a non-linear, feature-aligned, and irregular distribution of stipples.*

4.2.2 Contour-Hatching

To symbolize water movements, we developed a novel contour-hatching technique. Once a feature-aligned distance map was computed, individual stroke maps were irregularly placed with non-linear distance to shorelines to express motion. Similar to *Kalogerakis et al.* [2012], parameters were defined to provide artistic control over this placement:

- *Length* ($l \in \mathbb{R}^+$) defines the length of a stroke.
- *Thickness* ($t \in [0, 1]$) defines the width of a stroke.
- *Spacing* ($s \in [0, 1]$) controls the stroke density.
- *Randomness* ($r \in [0, 1]$) controls the stroke irregularity.



The main idea is to derive texture coordinates u, v for each rendering fragment by bilinearly sampling the distance maps D and T. To obtain the u-coordinate, waterline positions and the stroke width t were used to compute the fraction $u = (d - \varphi(d))/t$. To obtain the corresponding v-coordinate, feature-aligned distance values of the sampled distance map T were scaled by l to match the desired stroke length. In addition, noise was used to clamp the v-coordinate for an irregular placement of individual strokes, and the s and r parameters for density control and filtering. To render contour-hatches excessively wavy (P5), texture maps were used that had been digitized from hand-drawn strokes. Because the stroke placement works in object-space, it provides frame-to-frame coherence, and avoids the *shower door effect* known from techniques that work in image-space (e.g., [Kim et al. 2008b]).

From our analysis in Section 2, we observed that stroke layers of varying tone and density are used based on the shoreline distance (P4). This observation can be modeled by our algorithm using 3 layers with different parameter sets: dense, solid strokes near shorelines, loose strokes with irregular density, and strokes with shorter lengths near the medial axes (see Figure 13). Because our technique is texture-based, water movements can be modeled quite easily by shifting individual strokes along the major directions of the distance maps D or T, for instance by a temporal displacement of the v-coordinate using a sine function to animate rivers.

4.2.3 Water Vignetting and Cross-Hatching

In modern cartography, water vignettes are based on color gradients. A simple approach is to threshold the shoreline distance and interpolate between a shoreline and water body's color. We used this effect to complement our waterlining technique (Figure 1 left). A similar effect can be achieved by cross-hatching the shoreline areas by a tonal art map [Praun et al. 2001]. We used a map of five varying levels of stroke size and density, which were blended and mapped on water surfaces according to the shoreline distance, and parameterized according to the view distance to create a continuous level-of-abstraction [Semmo et al. 2012]. In contrast to tone-based shading, this approach does not affect shading of landmass (Figure 15), which allows us to visualize additional geospatial features, such as terrain.

4.2.4 Thematic Visualization

Figure 14: *Symbolizing flooded areas.*

We used distance maps for an automated, internal labeling that complies with the design principles of cartographers (P6/P7). The main idea is to derive piecewise cubic Bézier curves from the medial axes of the distance map D and warp text to these curves accordingly [Ropinski et al. 2007; Cipriano and Gleicher 2008]. To obtain the control points, pixels of the medial axes were traced and iteratively downsampled in image-space by nearest-neighbor interpolation. Together with the tangent information of the structure tensor, arc-length parameterization was used to warp text with the flow direction of water surfaces, and orient it with the viewing direction (Figure 16).

For symbolization, texture bombing was used and parameterized so that signatures always face the view direction when viewed in 3D (Figure 14) and comply with (P8). Alternatively, an example-based approach may be used to arrange signatures with more artistic control, for which we refer to the work by *Hurtut et al.* [2009].

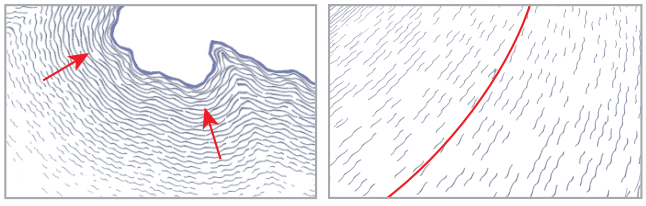


Figure 13: *Contour-hatching for water surfaces: (left) dense, wavy strokes near shorelines, (right) loose strokes near the medial axes.*

Figure 15: *A globe shaded by crosshatched strokes: (left) binary mask, (middle) result of [Webb et al. 2002], (right) our result.*

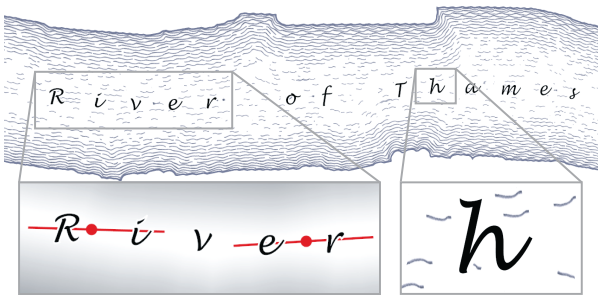


Figure 16: *Exemplary result of our labeling algorithm which aligns font glyphs according to the shoreline distance and orientation.*

5 Results

We have implemented our system using C++ and OpenGL/GLSL. OpenSceneGraph is used as the rendering engine to handle 3D data sets. The operations of the quantitative surface analysis are designed for parallel execution, and we have implemented them in CUDA to significantly improve overall performance. In particular, the PBA algorithm is used for the computation of Euclidean distance maps [Cao et al. 2010], together with a reduction for normalization [Nehab et al. 2011]. For text rendering, NVidia's `NV_path_rendering` extension is used to enable the rendering and transforming of high quality, instance-based text in a single pass. In the following, we demonstrate the usefulness and flexibility of our rendering techniques for different real-world data sets and potential usage scenarios.

5.1 Applications

Figure 17 shows a comparison of our rendering techniques using the example of *Spirit Lake* (at *Mount St. Helens*, USA). We observed that waterlining is a functional illustration technique that is able to communicate distance information quite effectively. The effects of water stippling and contour-hatching are similar but add a sense of motion and uncertainty. Coastal vignetting, by contrast, primarily focuses on the land-water interface itself to improve figure-ground perception. These techniques complement other cartography-oriented shading techniques quite well. This is demonstrated by shading the terrain in the environment with hachures of varying thickness according to the slope steepness [Buchin et al. 2004]. Moreover, our rendering techniques provide a level-of-abstraction (see the Appendix), which is exemplarily shown in the right image of Figure 17 where more or less waterlines are depicted according to the view distance to avoid visual clutter. The accompanying video demonstrates that this parameterization yields a smooth, continuous transition while zooming in and out. This example also demonstrates the ability of our system to handle 3D scenes and provide a spatial and temporal coherence. Because the rendering techniques are texture-based, they are independent from a model's geometric complexity. Finally, we experimented with using our rendering techniques concurrently. For instance, water stipples can be seamlessly blended with waterlines according to the view distance; or coastal vignetting to complement waterlining (Figure 1).

Our waterlining technique may be useful in flooding simulations, i.e., to assess distances to the nearest safety zones for evacuation planning. When performed over time, waterlines dynamically shift with the flood distribution to convey motion and enhance the depiction of land cover. This effect is demonstrated in the supplementary video and is exemplarily shown in Figure 18 for the city of Boston (USA). Here, a plane is used and temporally shifted upwards to represent the change in the mean sea level. Using this plane as a clipping mask with an orthographic projection, the corresponding flooded areas are obtained and shaded in real-time.

Table 1: *Performance evaluation (in* ms*): distance / feature-aligned distance transform (D/T), orientation and medial axis computation.*

Image Res.	D	T	Orient.	M. Axes	Total
128×128	1.6	26.6	0.1	0.7	29.0
256×256	2.2	96.3	0.2	0.8	99.5
512×512	4.5	371.6	0.6	0.9	377.6

Table 2: *Performance evaluation of our illustrative rendering techniques for different screen resolutions (in frames-per-second).*

Screen Res.	waterlining	stippling	contour-hatching
800×600	534	162	159
1280×720	523	88	84
1600×900	521	59	55
1920×1080	514	42	41

Figure 19 demonstrates the usefulness of our rendering techniques in urban planning and cultural heritage. Within these domains, it is often desired to avoid authentic impressions, in particular because of missing evidence in the (re)construction or because construction plans may be altered in the future. The top image shows a topographical reconstruction of ancient Cologne, in which contour-hatching is used to express uncertainty. We animated the individual strokes to express water movements. In addition, symbolization is used to highlight those river areas that were flooded in ancient times. The bottom image shows a bridge construction, where water stippling is used to add expressiveness. As can be seen, the stipple density is increased in the shadowed areas to improve depth perception.

5.2 Performance Evaluation

The performance tests of our system were conducted on an Intel® Xeon™ $4\times$ 3.06 GHz with 6 GByte RAM and NVidia® GTX 660 Ti GPU with 2 GByte VRAM. We used *Spirit Lake* (Figure 17) as a test model. The results in Table 1 show the run-time of the quantitative surface analysis scales with the resolution of a distance map. The feature-aligned distance transform is shown to be a limiting factor; however, its implementation is not heavily optimized and we see potential to increase the performance. We compared different sizes of distance maps for our illustrative rendering techniques. Similar to signed distance maps [Green 2007], we achieve stable results when bilinearly sampling a low resolution of a feature-aligned distance map (128×128 pixels). Note that the timings for the distance maps include normalization. Table 2 shows that our illustrative rendering techniques perform at real-time frame-rates in HD resolution. During rendering, we observe that our shading techniques are fill-limited and achieve, in SD resolution, twice the performance of an HD resolution. We conclude that our system for feature-aligned waterlining, stippling, and hatching performs in real-time, and therefore is applicable to render animated 3D scenes.

5.3 Limitations

Our shading techniques work in object-space and the computation of distance maps requires closed polygons for processing. For large-scale water surfaces with complex courses, distance maps of high resolution are required to achieve quality shading results. Here, memory resources limit the distance map sizes that can be processed by a GPU; for our system $\approx$ 16 Megapixels (MP) with 2 GByte VRAM. Moreover, we observed that our feature-aligned distance transform does not perform in real-time when computing distance maps with > 0.25 MP. Nonetheless, we observed that distance maps of 256×256 pixels are sufficient for most water surfaces when using bilinear filtering.

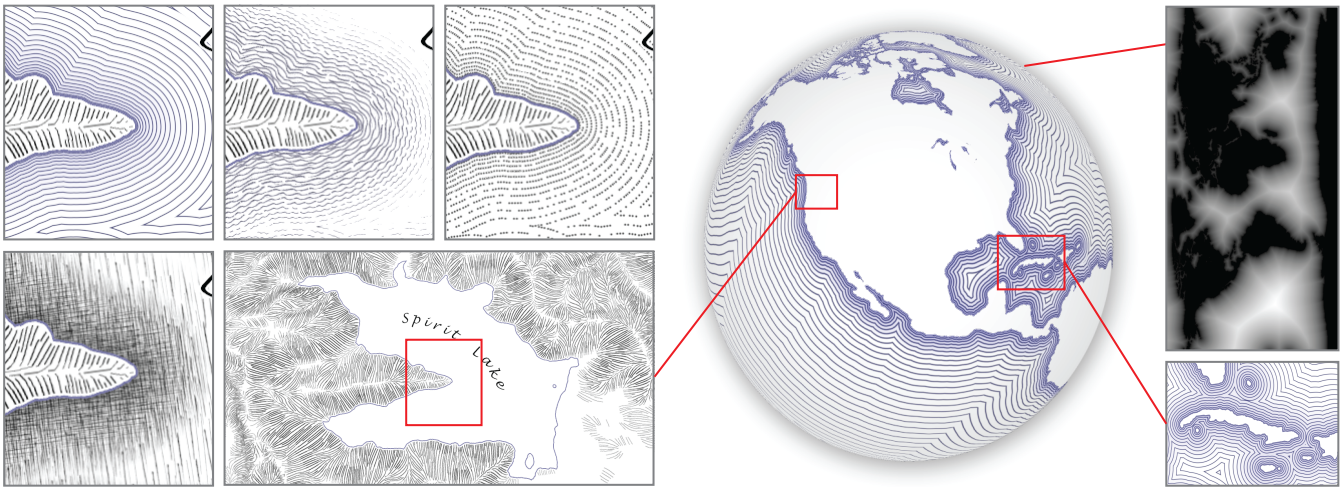


Figure 17: *Exemplary results of our rendering techniques for 3D mapping, (left) compared with each other within the environment of Mount St. Helens, (right) waterlining applied to a globe that provides a continuous level-of-abstraction when zooming in and out.*

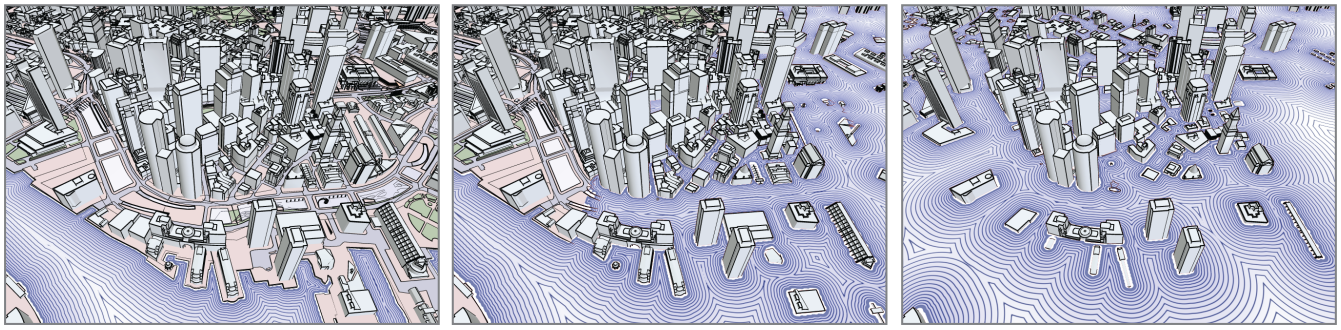


Figure 18: *Flooding simulation for the city of Boston enhanced by our waterlining technique and illustrative rendering to express uncertainty.*

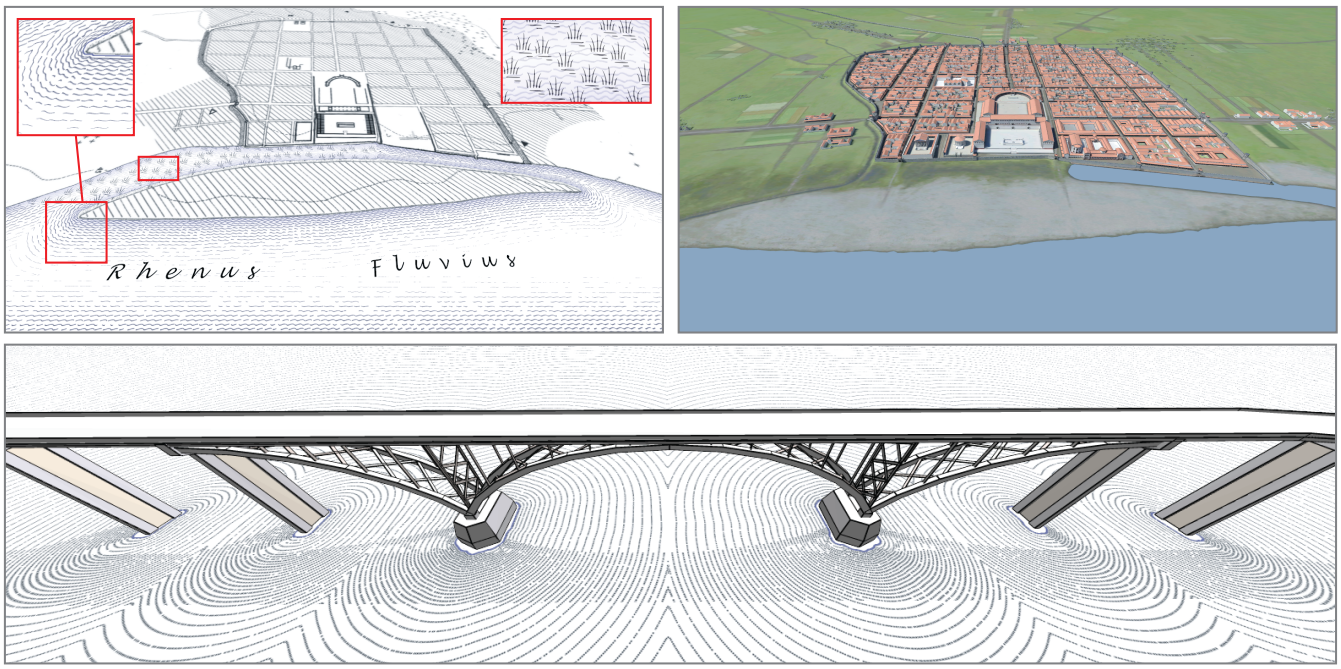


Figure 19: *Further results of our rendering techniques for urban planning and landscaping: (top) contour-hatching used to express uncertainty in a reconstructed topographical model of ancient Cologne (Germany), which served as the basis for the 3D reconstruction shown to the right, (bottom) water stippling used to enhance the rendering of a bridge construction.*

6 Conclusions and Future Work

We present a system for rendering water surfaces with cartography-oriented design. Our real-time rendering techniques adopt design principles from traditional cartography to improve figure-ground perception and express a sense of motion. For contour-hatching, we propose a novel feature-aligned distance transform to align individual strokes with the shorelines of water surfaces. Results show that our techniques provide temporal and spatial coherence, can be parameterized for a view-dependent level-of-abstraction, and can be useful within 3D geovirtual environments for map exploration, urban planning, landscaping, and disaster management. Because of their application to geospatial data, we plan to elaborate on the usefulness of our techniques for geovisualization. Further, we plan to conduct a user study to confirm significant effects in orientation, navigation, and analysis tasks performed within 3D geovirtual environments.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments. This work was partly funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group "4DnDVis", and was partly supported by the ERC-2010-StG 259550 XSHAPE grant.

References

BRATKOVA, M., SHIRLEY, P., AND THOMPSON, W. B. 2009. Artistic rendering of mountainous terrain. *ACM Trans. Graph. 28*, 102:1–102:17.

BROX, T., BOOMGAARD, R., LAUZE, F., WEIJER, J., WEICKERT, J., MRÁZEK, P., AND KORNPROBST, P. 2006. Adaptive Structure Tensors and their Applications. *Visualization and Processing of Tensor Fields*, 17–47.

BUCHIN, K., SOUSA, M. C., DÖLLNER, J., SAMAVATI, F., AND WALTHER, M. 2004. Illustrating Terrains using Direction of Slope and Lighting. In *ICA Mountain Carthography Workshop*, 259–269.

CAO, T.-T., TANG, K., MOHAMED, A., AND TAN, T.-S. 2010. Parallel Banding Algorithm to compute exact distance transform with the GPU. In *Proc. I3D*, 83–90.

CHRISTENSEN, A. H. 1999. Cartographic Line Generalization with Waterlines and Medial-Axes. *Cartography and Geographic Information Science 26*, 1, 19–32.

CHRISTENSEN, A. H. 2008. A Reflection on the Waterlining Technique in Relation to the History of Map Ornamentation. *The Cartographic Journal 45*, 1, 68–78.

CIPRIANO, G., AND GLEICHER, M. 2008. Text Scaffolds for Effective Surface Labeling. *IEEE Trans. Vis. Comput. Graphics 14*, 6, 1675–1682.

COCONU, L., DEUSSEN, O., AND HEGE, H. 2006. Real-time pen-and-ink illustration of landscapes. In *Proc. NPAR*, 27–35.

DANIELSSON, P.-E. 1980. Euclidean distance mapping. *Computer Graphics and Image Processing 14*, 3, 227–248.

DARLES, E., CRESPIN, B., GHAZANFARPOUR, D., AND GONZATO, J. 2011. A Survey of Ocean Simulation and Rendering Techniques in Computer Graphics. *Comput. Graph. Forum 30*, 1, 43–60.

DEUSSEN, O., AND STROTHOTTE, T. 2000. Computer-generated pen-and-ink illustration of trees. In *Proc. ACM SIGGRAPH*, 13–18.

DÖLLNER, J., AND WALTHER, M. 2003. Real-time expressive rendering of city models. In *Proc. IEEE IV*, 245–250.

EDEN, A. M., BARGTEIL, A. W., GOKTEKIN, T. G., EISINGER, S. B., AND O'BRIEN, J. F. 2007. A Method for Cartoon-Style Rendering of Liquid Animations. In *Proc. ACM Graphics Interface*, 51–55.

FRENCH, T. 1918. *A manual of engineering drawing for students and draftsmen*. McGraw-Hill book company.

FRISKEN, S. F., PERRY, R. N., ROCKWOOD, A. P., AND JONES, T. R. 2000. Adaptively sampled distance fields: a general representation of shape for computer graphics. In *Proc. ACM SIGGRAPH*, 249–254.

GERL, M., AND ISENBERG, T. 2013. Interactive example-based hatching. *Computers & Graphics 37*, 1-2, 65–80.

GIRSHICK, A., INTERRANTE, V., HAKER, S., AND LEMOINE, T. 2000. Line direction matters: an argument for the use of principal directions in 3D line drawings. In *Proc. NPAR*, 43–52.

GLANVILLE, R. S. 2004. Texture Bombing. In *GPU Gems*. Addison-Wesley, 323–338.

GÖTZELMANN, T., ALI, K., HARTMANN, K., AND STROTHOTTE, T. 2005. Form Follows Function: Aesthetic Interactive Labels. In *Proc. CAe*, 193–200.

GREEN, C. 2007. Improved alpha-tested magnification for vector textures and special effects. In *ACM SIGGRAPH Courses*, 9–18.

HUFFMAN, D. P. 2010. On Waterlines: Arguments for their Employment, Advice on their Generation. *Cartographic Perspectives, NACIS 66*, 23–30.

HURTUT, T., LANDES, P.-E., THOLLOT, J., GOUSSEAU, Y., DROUILLHET, R., AND COEURJOLLY, J.-F. 2009. Appearance-guided Synthesis of Element Arrangements by Example. In *Proc. NPAR*, 51–60.

IMHOF, E. 1972. *Thematische Kartographie*, vol. 10. Walter de Gruyter.

IMHOF, E. 1975. Positioning names on maps. *The American Cartographer 2*, 2, 128–144.

ISENBERG, T. 2013. Visual Abstraction and Stylisation of Maps. *The Cartographic Journal 50*, 1, 8–18.

JOBST, M., AND DÖLLNER, J. 2008. 3D City Model Visualization with Cartography-Oriented Design. In *Proc. REAL CORP*, 507–516.

KALOGERAKIS, E., NOWROUZEZAHRAI, D., BRESLAV, S., AND HERTZMANN, A. 2012. Learning hatching for pen-and-ink illustration of surfaces. *ACM Trans. Graph. 31*, 1, 1:1–1:17.

KIM, D., SON, M., LEE, Y., KANG, H., AND LEE, S. 2008. Feature-guided Image Stippling. *Comput. Graph. Forum 27*, 4, 1209–1216.

KIM, Y., YU, J., YU, X., AND LEE, S. 2008. Line-art illustration of dynamic and specular surfaces. *ACM Trans. Graph. 27*, 5, 156:1–156:10.

KIM, S., WOO, I., MACIEJEWSKI, R., AND EBERT, D. 2010. Automated Hedcut Illustration Using Isophotes. In *Proc. Smart Graphics*, 172–183.

KRAAK, M., AND ORMELING, F. 2003. *Cartography: Visualization of Geospatial Data*. Pearson Education.

KYPRIANIDIS, J. E., COLLOMOSSE, J., WANG, T., AND ISENBERG, T. 2012. State of the 'Art': A Taxonomy of Artistic Stylization Techniques for Images and Video. *IEEE Trans. Vis. Comput. Graphics 19*, 5, 866–885.

MAASS, S., AND DÖLLNER, J. 2008. Seamless Integration of Labels into Interactive Virtual 3D Environments Using Parameterized Hulls. In *Proc. CAe*, 33–40.

MACEACHREN, A. 1995. *How Maps Work*. Guilford Press.

MERIAN, M. 2005. *Topographia Germaniae*.

NEHAB, D., MAXIMO, A., LIMA, R. S., AND HOPPE, H. 2011. GPU-efficient recursive filtering and summed-area tables. *ACM Trans. Graph. 30*, 176:1–176:12.

PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2000. Lapped textures. In *Proc. ACM SIGGRAPH*, 465–470.

PRAUN, E., HOPPE, H., WEBB, M., AND FINKELSTEIN, A. 2001. Real-time hatching. In *Proc. ACM SIGGRAPH*, 581–586.

ROBINSON, A. H., MORRISON, J. L., MUEHRCKE, P. C., KIMERLING, A. J., AND GUPTILL, S. C. 1995. *Elements of cartography*. New York: John Wiley & Sons.

RONG, G., AND TAN, T.-S. 2006. Jump flooding in GPU with applications to Voronoi diagram and distance transform. In *Proc. ACM I3D*, 109–116.

ROPINSKI, T., PRASSNI, J.-S., ROTERS, J., AND HINRICHS, K. 2007. Internal Labels as Shape Cues for Medical Illustration. In *Proc. VMV*, 203–212.

SALISBURY, M. P., WONG, M. T., HUGHES, J. F., AND SALESIN, D. H. 1997. Orientable textures for image-based pen-and-ink illustration. In *Proc. ACM SIGGRAPH*, 401–406.

SEMMO, A., TRAPP, M., KYPRIANIDIS, J. E., AND DÖLLNER, J. 2012. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. *Comput. Graph. Forum 31*, 885–894.

TYNER, J. 2010. *Principles of map design*. Guilford Press.

WEBB, M., PRAUN, E., FINKELSTEIN, A., AND HOPPE, H. 2002. Fine tone control in hardware hatching. In *Proc. NPAR*, 53–58.

WEI, L., LEFEBVRE, S., KWATRA, V., TURK, G., ET AL. 2009. State of the Art in Example-based Texture Synthesis. In *Eurographics 2009 State of the Art Report*, 93–117.

XU, K., COHEN-OR, D., JU, T., LIU, L., ZHANG, H., ZHOU, S., AND XIONG, Y. 2009. Feature-aligned shape texturing. *ACM Trans. Graph. 28*, 108:1–108:7.

YU, J., JIANG, X., CHEN, H., AND YAO, C. 2007. Real-time cartoon water animation. *Computer Animation and Virtual Worlds 18*, 4-5, 405–414.

Appendix

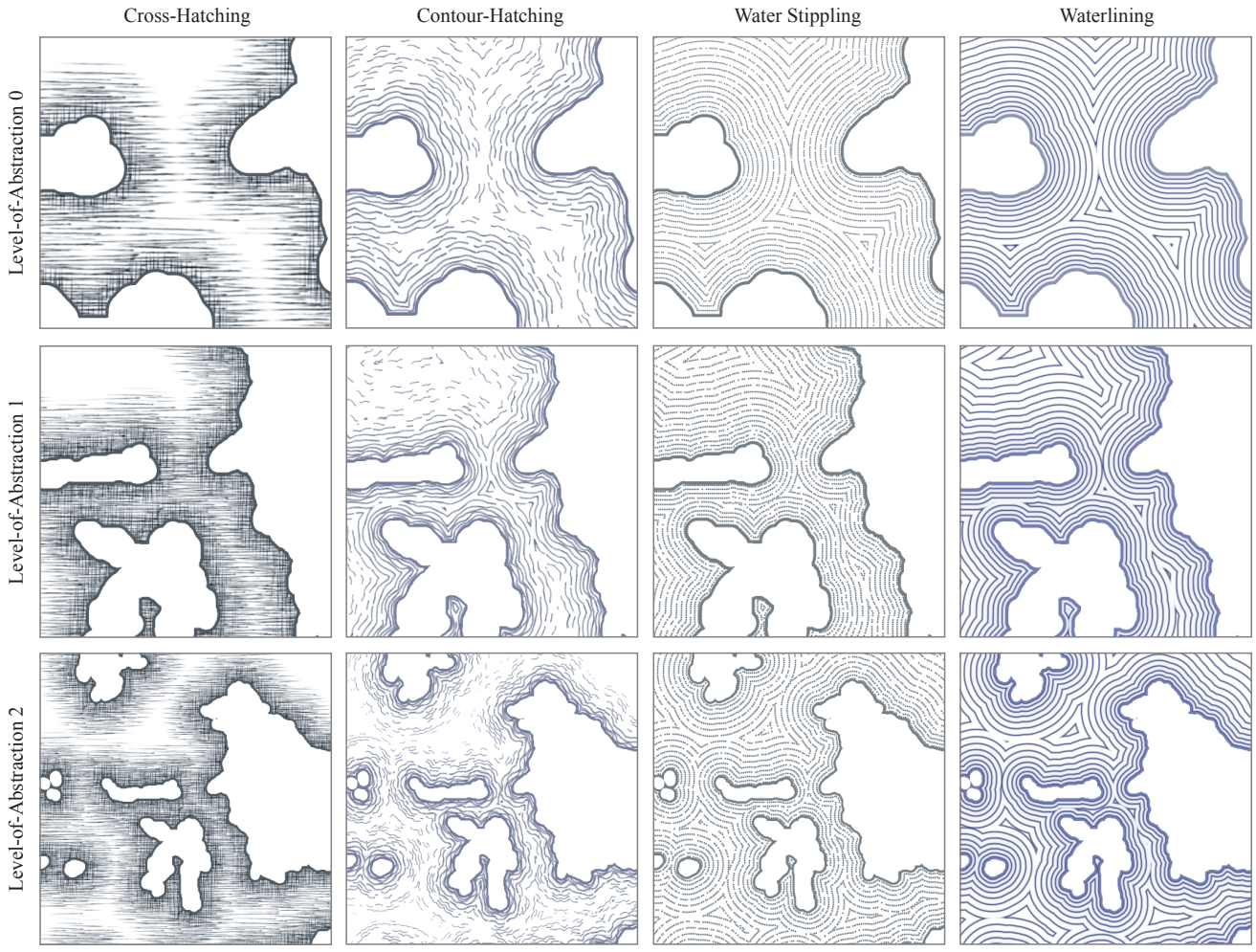


Figure 20: *Exemplary parameterization results of our rendering techniques for a view-dependent level-of-abstraction. In order to reduce visual clutter at high view distances (bottom row), the number of rendered waterlines, stipples and hatches are reduced significantly. Once these parameterizations have been authored by our system, blending between them is performed in real-time using OpenGL fragment shaders.*