Interactive Image Filtering for Level-of-Abstraction Texturing of Virtual 3D Scenes *

Amir Semmo Jürgen Döllner[1]

Hasso Plattner Institute, University of Potsdam, Germany

Abstract

Texture mapping is a key technology in computer graphics. For the visual design of 3D scenes, in particular, effective texturing depends significantly on how important contents are expressed, e.g., by preserving global salient structures, and how their depiction is cognitively processed by the user in an application context. Edge-preserving image filtering is one key approach to address these concerns. Much research has focused on applying image filters in a post-process stage to generate artistically stylized depictions. However, these approaches generally do not preserve depth cues, which are important for the perception of 3D visualization (e.g., texture gradient). To this end, filtering is required that processes texture data coherently with respect to linear perspective and spatial relationships. In this work, we present an approach for texturing 3D scenes with perspective coherence by arbitrary image filters. We propose *decoupled deferred texturing* with (1) caching strategies to interactively perform image filtering prior to texture mapping and (2) for each mipmap level separately to enable a progressive level of abstraction, using (3) direct interaction interfaces to parameterize the visualization according to spatial, semantic, and thematic data. We demonstrate the potentials of our method by several applications using touch or natural language inputs to serve the different interests of users in specific information, including illustrative visualization, focus+context visualization, geometric detail removal, and semantic depth of field. The approach supports frame-to-frame coherence, order-independent transparency, multitexturing, and content-based filtering. In addition, it seamlessly integrates into real-time rendering pipelines, and is extensible for custom interaction techniques.

Keywords: image filtering, level of abstraction, texturing, virtual 3D scenes, visualization, interaction, focus+context interfaces

1. Introduction

Common 3D scenes are characterized by their visual complexity. In particular, the appearance of most objects can be defined by textures used to design the objects' appearance. Prominent examples of texture-based 3D scenes include virtual 3D building, city, and landscape models or environments used for gaming applications. Texture mapping is a key technology in today's graphics hardware for the visual design of 3D scenes [1]. In particular, it provides important functionalities used in both photorealistic and non-photorealistic real-time image synthesis. Common texture maps encode color, diffuse, normal, or displacement information as *surface properties* to enrich shading and lighting effects (e.g., for the building façades in Figure 1). Rendering these properties in detail, however, does not automatically lead to high image quality such as with respect to the effectiveness of the information transfer regarding the user [2]. For example, in non-phototealistic rendering (NPR), approaches particularly take into account a user's background, task, and perspective view, facilitating the expression, recognition, and communication of important or prioritized information [3, 4, 5, 6].

To improve image quality of textured 3D scenes, important contents of textures should be emphasized while less significant details should be removed—a challenging task because feature contours and global salient structures must be preserved. A promising approach to address this problem is edge-preserving image filtering. Popular image filters serve as smoothing or enhancing operators such as the bilateral filter [7] and difference of Gaussians [8]. Previous works have focused on applying these filters in a post-process stage on the rendered results to foster an artistically stylized rendering [9]. These approaches are able to smooth low-contrast regions and preserve high-contrast edges; however, they are generally not able to preserve depth cues important for perceiving model contents as three-dimensional (e.g., occlusion, texture gradient). For instance, the fine granular patterns on the ground and rooftops of the 3D scene shown in Figure 1 are not preserved because spatial relationships and linear perspective are not considered, and fine objects may become indistinguishable from the background. For effective visual information encoding, "good image filtering" needs to preserve these cues to help perceive relative positions, sizes, distances, and shapes more clearly [10, 11].

This work explores *level-of-abstraction* (LoA) texturing and its interactive parameterization by means of image filtering, which refers to adapting the spatial granularity at which 3D scene contents should be represented [12]. Our contributions are based on the following key aspects:

(1) Filtering should be perspectively coherent. Linear perspective, occlusion, and texture gradient are effective cues for humans to infer depth [13, 14]. Texture mapping considers these cues in perspective projections by foreshortening and scaling.

[1] {first.last}@hpi.de | http://www.hpi3d.de

This is the authors' version of the work. The definitive version will be published in Computers & Graphics, 2015. doi: 10.1016/j.cag.2015.02.001.

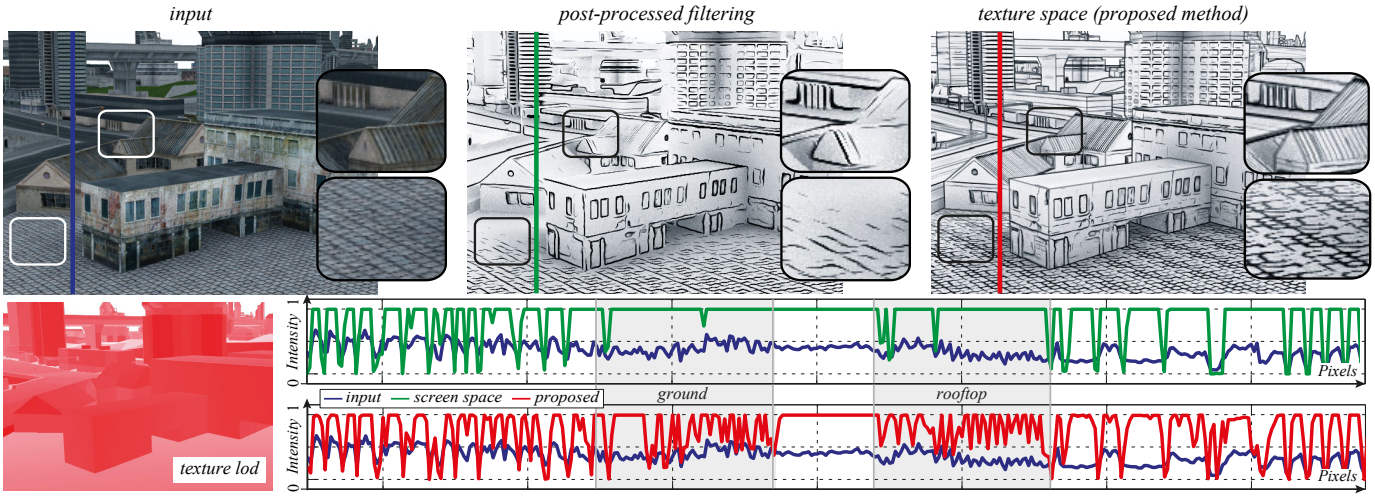

Figure 1: Exemplary results of a flow-based difference-of-Gaussians filtering (FDoG) for a textured 3D scene, rendered in real-time using our framework. The closeups and scanline plots illustrate the high accuracy for texture gradients when using our proposed method (red line) instead of conventional filtering in a post-process stage on the rendered color image (green line).

Therefore, our approach performs image filtering prior to texture mapping via decoupled deferred texturing. It is a simple yet effective method of preserving these cues without requiring modifications of the original filter algorithms.

(2) Filtering should be interactively parameterized. To serve the different interests of users in prioritized information, the LoA should be dynamically adapted to a user's context, which requires specialized interaction techniques for parameterization. To this end, the proposed approach is coupled with an extensible interaction framework to parameterize a context-aware image filtering according to spatial, semantic, and thematic data. The framework provides intuitive interfaces such as touch-based or natural language inputs via textual descriptions. In addition, interactive frame rates should be maintained to provide a responsive system during interaction and navigation in 3D space (e.g., dynamic viewing situations, regions of interest). For this reason, we contribute per-fragment and progressive filtering together with caching strategies to enable real-time frame rates for local image filters without requiring to pre-process texture data.

In image-based artistic rendering it is often desired to give the impression that texture features have been applied (painted) on a flat canvas [15]. Instead, this paper draws upon the potentials of image filtering for visualization purposes including cognitive principles (here: psychological design aspects) and direct user interaction for parameterization; two fields of research that pose contemporary challenges for the NPR community [16]. Accordingly, we want to preserve a perspective-coherent scale of texture features, an approach also practiced by artists to enhance depth sensation (Figure 2). The benefits are shown for applications such as focus+context visualization, illustrative visualization, and geometric detail removal. Because filtering is performed in texture space, however, no explicit geometric abstraction is applied, for which specialized rendering techniques are required.

This paper represents an extended journal version of the

Figure 2: The painting "Paris Street, Rainy Day" (1877) by Gustave Caillebotte (source: Google Art Project). The artist carefully uses texture gradient with linear perspective to enhance depth sensation. Notice how the cobblestones get progressively smaller in depth until they are completely smooth.

CAe 2014 paper by Semmo and Döllner [17]. Besides a significant revision of all sections, we particularly expand on the fields of user interaction for NPR. With respect to this, the following three major revisions have been made: (1) Section 3 expands our review of related work, now including topics such as interaction design and user interfaces for 3D virtual environments, (2) the new Section 5 proposes an integrated, extensible interaction framework to intuitively parameterize our methods, and (3) Section 6.2 expands our original applications by using touch and natural language interfaces, where we demonstrate the versatile application of our framework to geospatial tasks, including exploration, navigation, and orientation. Further, we expand Section 6.2 on visualization results and applications to provide a more elaborate discussion of our techniques with respect to an effective 3D information transfer. Furthermore, the new Section 6.3 provides a quantitative analysis of visual clutter reduction and visual saliency to evaluate the effect of our image filtering methods for focus+context visualization. Finally, Section 7 provides an updated prospect on future work.

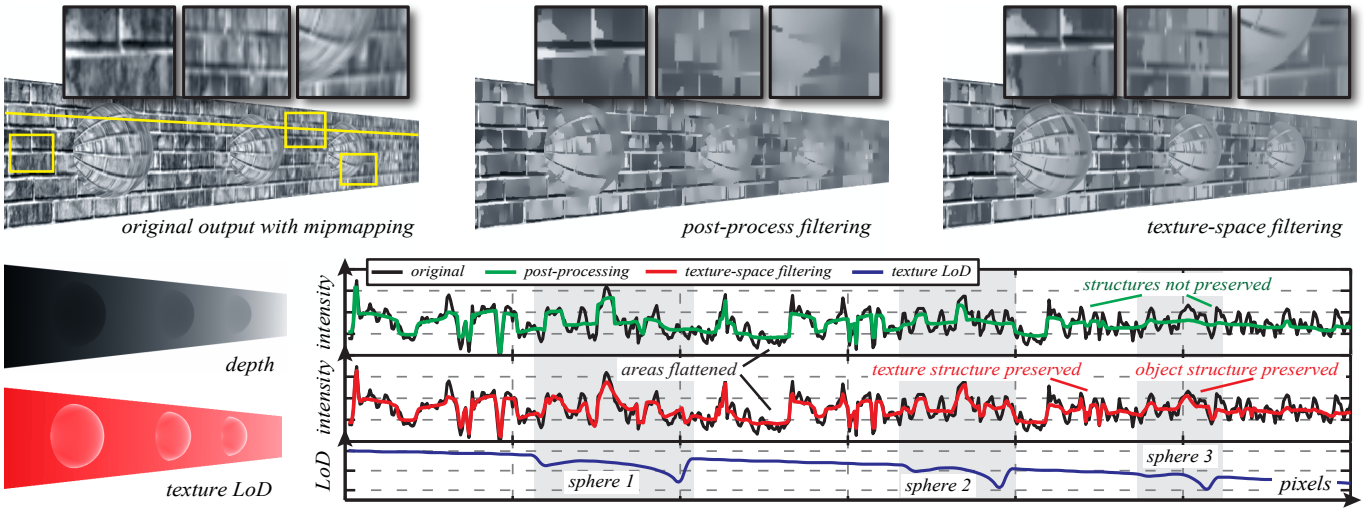

Figure 3: Exemplary textured 3D scene for which the flow-based bilateral filter ($\rho = 2.0, n_e = 1, n_a = 4, \sigma_d = 4.0, \sigma_r = 5.0$) is applied: (1) on the original output, (2) on each mipmap level prior to texture mapping. The scanline plots illustrate the differences in regions of high texture LoD and occlusion, where the second approach clearly preserves texture gradients and object boundaries.

The remainder of this extended paper is structured as follows. Section 2 explains the relevance of perspective coherence for image filtering in 3D perspective views. Section 3 reviews related work on image filtering, non-photorealistic rendering, effective 3D information transfer, and user interaction. Section 4 presents the technical approach to deferred texture filtering, parameterized via our extensible interaction framework presented in Section 5. Applications for our methods together with interaction techniques are presented, discussed, and evaluated in Section 6. Finally, Section 7 concludes this paper.

2. Background – Why Perspective Coherence Matters

This section motivates *perspective coherence* for LoA texturing. The example 3D scene shown in Figure 3 is based on a flow-based bilateral filter [18]. The input textures were converted to a low-pass filtered mipmap pyramid [19] used for antialiasing. The filtering was performed (1) in a post-process stage on the rendered results and (2) in texture space prior to perspective projection. Given a texture $T : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ within the RGB color space as input, the first approach is described by an image filtering F performed *after* sampling the convolution of $T * P$ via mipmapping, with P being a projector function that maps texture information into the domain of the rendered image $I : \mathbb{R}^2 \rightarrow \mathbb{R}^3$ and K being the mipmap kernel:

$$I(x, y) = F[K \cdot (T * P)](x, y). \quad (1)$$

While this approach works in a similar manner as pre-filtering a texture in regions of high texture level of detail (LoD), it is not able to preserve structures in regions of low texture LoD because only compressed information serves as input for the image filtering kernel F. This effect can be observed in the intensity plots along the yellow scanline shown in Figure 3 and in the conceptual overview depicted in Figure 4 (top row). To preserve depth cues, instead, image filtering requires access to high-frequency texture information prior to perspective projection.

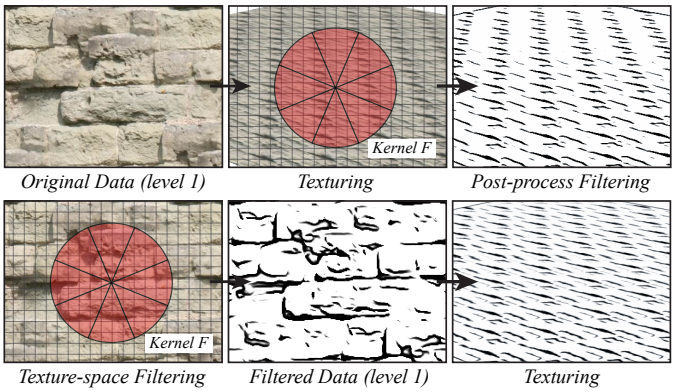

Figure 4: Comparison between post-process filtering (top row) and texture-space filtering (bottom row) using the FDoG filter [18], with mipmapping enabled for both approaches.

In the second approach, mipmap levels are filtered separately prior to texture mapping to approximate the filtering effect of F on $K(T * P)$, with K_F being the mipmap kernel that uses the pre-filtered mipmap levels:

$$I'(x, y) = K_F \cdot (T * P)(x, y). \quad (2)$$

Figure 3 (scanline plots) and Figure 4 (bottom row) demonstrate that this approach naturally preserves texture gradients and object boundaries when trilinearly filtering the respective mipmap levels. While this is not a new approach in 3D computer graphics (e.g., used for pre-filtering shadow maps [20]), artists use similar principles in their work to enhance the sensation of depth by adapting the detail and size of texture features with the linear perspective (Figure 2).

We contribute an interactive framework that implements Equation 2 with optimization techniques to enable real-time performance for local image filters. The framework uses conventional mipmapping and projective texturing capabilities of

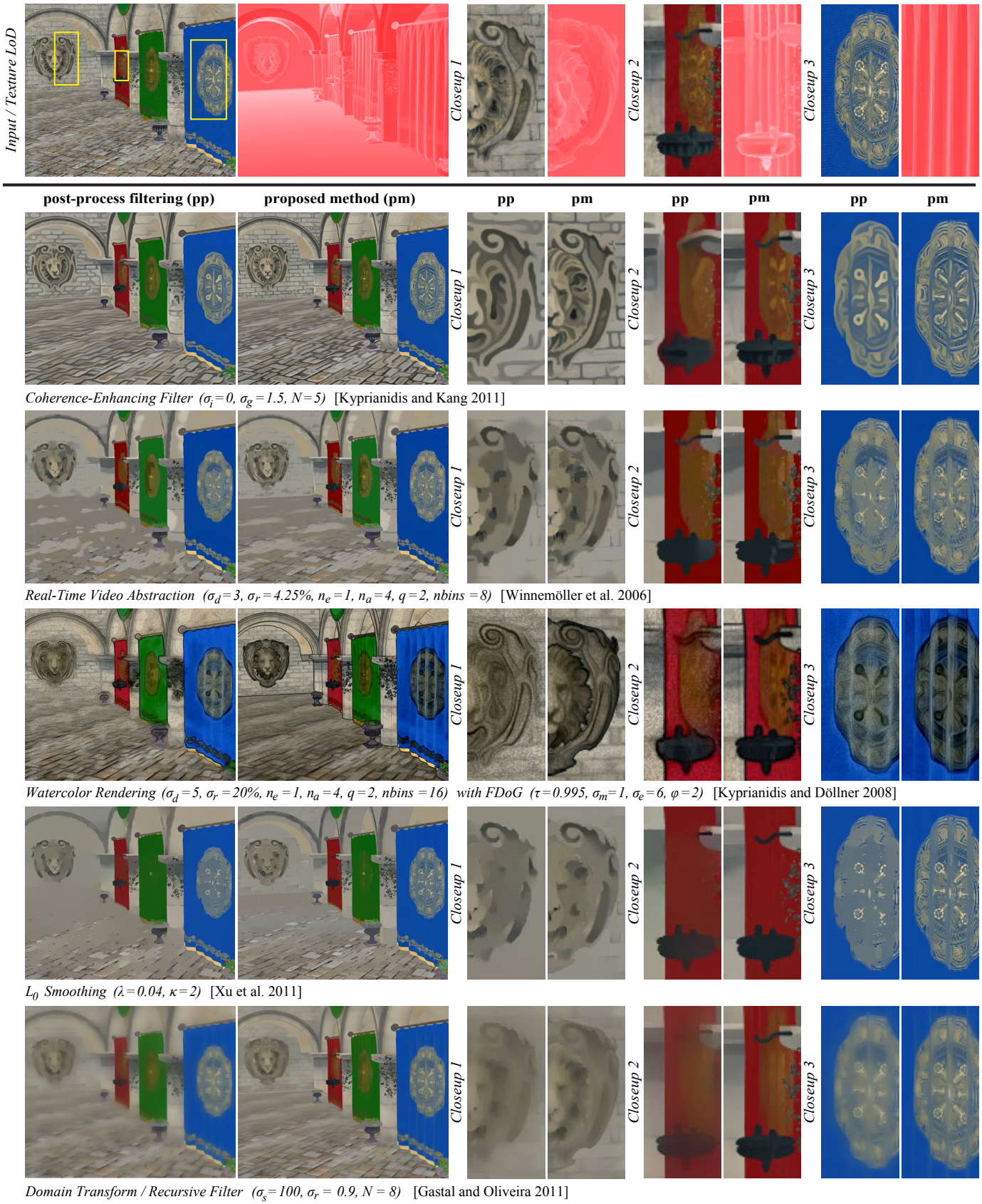

Figure 5: Image filters applied to a 3D scene using our system. The top row shows the original rendering output and texture LoD, the first column the results of post-process filtering (pp), and the second column our proposed method (pm). The texture gradients on the floor, the textured banners, and the lion figure in the back are aggressively smoothed when filtered in a post-process stage, while our approach preserves their structures and overall object borders without compromising the filters' qualities. (Sponza Atrium scene © Marko Dabrovic and Frank Meinl from Crytek. All rights reserved.)

the hardware-accelerated rendering pipeline to inherently take the linear perspective, occlusions, and texture gradients into account. Using our approach with state-of-the-art edge-preserving image filters, we performed a comparison between the filtering approaches (1) and (2). Results are presented in Figure 5 and the accompanying video. The comparison was carried out qualitatively on the basis that texture gradients and occlusions induced by linear perspective are preserved. The results demonstrate how each filter's capability to preserve texture gradients and occlusions is improved, while the filters' qualities with respect to image smoothing or enhancing remain unaffected. For instance, the ringing artifacts of the red banner in Figure 5 are reduced with the coherence-enhancing filter [21], the texture gradient on the floor and its transition to the back wall are preserved with bilateral filtering [22] and L_0 gradient minimization [23], and the overall structures are emphasized with much higher precision.

3. Related Work

Related work is found in the fields of edge-preserving image filtering, texture mapping and filtering for 3D scenes, focus+context visualization, and interaction in 3D virtual environments.

3.1. Edge-preserving Image Filtering

Edge-preserving filtering emerged as an essential building block to reduce image details without loss of salient structures. Many filters have been proposed and explored using image abstraction and highlighting [9], and thus are of major interest for our work. Typical approaches operate in the spatial domain, use a kind of anisotropic diffusion [24], and are designed for parallel execution. We have implemented a range of local filters in our prototype to demonstrate how they can be used for real-time filtering of 3D scene contents for progressive LoA texturing. A popular choice is the bilateral filter, which works by weight averaging pixel colors in a local neighborhood based on their distances in space and range [7]. This method has been used for real-time image-based artistic rendering [22] and enhanced by flow-based implementations [18, 25] adapted to the local image structure to provide smooth outputs at curved boundaries. Because the bilateral filter may fail when used in high-contrast images, we also explored the usage of the anisotropic Kuwahara filter [26] to maintain a uniform LoA due to local area flattening, and coherence-enhancing filtering based on directional shock filtering [21] (see Figure 5). Because our approach is designed for generic application without requiring modifications of the original algorithms, future extensions are easy to implement. Additional categories include mean-shift filtering for discontinuity-preserving smoothing [27] and saliency-guided image abstraction [28], morphological filtering based on dilation and erosion (e.g., for watercolor rendering [29]), and geodesic filtering using distance transforms [30, 31].

Edge detection is based on finding zero-crossings or thresholding the gradient magnitude of an image. A popular choice is the difference-of-Gaussians (DoG) filter [8] and its enhanced flow-based variants [18, 32, 33] to create smooth coherent outputs for line and curve segments. Complemented by an image-space edge detection using geometry information [34], we used the DoG filter to enhance important edges based on both geometry and texture information. Our method produces accurate filtering results with respect to linear perspective to preserve texture gradients (Figure 1), performs in real-time, and provides frame-to-frame coherence.

Recent filters focused on image decompositions by using optimization methods such as weighted least squares [35], local extrema for edge-preserving smoothing [36], locally weighted histograms [37], domain transforms [38], L_0 gradient minimizations [23], guided filtering [39], region covariances [40], and, most lately, modified bilateral filtering [41]. Although most of them do not provide interactive frame rates, we have implemented GPU versions of a variety of these algorithms to demonstrate how they can serve perspective-coherent filtering of diffuse, normal, and ambient occlusion maps for geometric detail removal and illustrative visualization.

3.2. Texture Mapping and Filtering for 3D Scenes

Using texture-based methods for coherent stylization is not a new approach. For an overview on this topic we refer to the survey by Bénard et al. [15]. Relevant object-space methods reduced perspective distortions and scale variations via mipmapping (i.e., art maps [42, 43]) and infinite zoom mechanisms [44] to maintain a quasi-constant size of texture elements for arbitrary view distances. Conceptually, our approach behaves very similar to the art maps approach but generalizes the idea of using mipmapping for coherent stylization with respect to image filtering in two ways: (1) instead of using pre-designed mip-map levels or relying on procedural modeling our approach interactively filters high-detail texture information at run-time (e.g., photorealistic imagery), (2) our approach enables a user-defined LoA texturing (e.g., for focus+context visualization) that is not limited to color maps but also copes with multiple layers of textures used for visualization (e.g., normal maps, thematic maps). Previous approaches used bilateral, DoG, and Kuwahara filtering with *G-buffer* information [45] in a post-process stage to express uncertainty [46], direct a viewer's gaze to certain image regions [6, 47], and convey different emotional and experiential representations [48]. Our approach also uses *G-buffer* information for deferred rendering, but decouples filtering from shading to preserve texture gradients and object boundaries when mapping the filtered results.

Previous work has supported the assertion that perspective is commonly used as an important source of depth information [10, 13, 14]. In particular, surface textures are essential components of 3D scenes to judge distances, shapes, and spatial layouts [2, 49]. A large body of research is dedicated to the way human sensory systems process these pictorial depth cues [50]; however, there are only a few works that reflect the importance of preserving them when filtering information. A recent study [51] showed that using the results of a DoG filter on diffuse maps significantly improved depth perception in thematic 3D visualization. We demonstrate similar approaches to edge enhancement, but with a filtering that can be interactively parameterized at run-time.

To control the detail of 3D shapes via parameterized lighting, major related work is found in cartoon shading [52, 53]

that supports view-dependent effects (e.g., LoA, depth of field). We propose LoA texturing of photorealistic diffuse maps as an orthogonal approach to these works. In addition, we demonstrate how salient structures of textures can be preserved using flow-based image filters for stylized shading and lighting.

3.3. Focus+Context Visualization

LoA texturing, as is proposed in this work, provides effective means for focus+context visualization. *Focus+context* describes the concept to visually distinguish between important or relevant information from closely related information [54]. Focus+context visualization conforms with the visual information-seeking mantra [55] by enabling users to interactively change the visual representation of data for points and regions of interest using highlighting techniques while maintaining a context for orientation guidance, i.e., to solve the problem of over-cluttered visual representations. It has the potential to improve the perception of prioritized information [3, 28] and direct a viewer's focus to certain image regions [4]. A common method is to parameterize image filters according to view metrics (e.g., view distance [47]) or by explicitly defined regions of interest [4, 56] to select and seamlessly combine different LoA representations of 3D scene contents [12]. In this paper, we demonstrate our interactive approach on several focus+context examples. In particular, our methods are able to enhance several applications by incorporating texture information coherently with respect to linear perspective, including the *stylized focus pull* [4] and *semantic depth of field* [57] effects for information highlighting and abstraction [58].

3.4. Interaction in 3D Virtual Environments

Many systems use a classical mouse/keyboard setup with a graphical user interface to help inspect and parameterize a visualization of 3D scene contents [59]. Direct manipulation is typically performed via ray casting to determine intersections of a pointing device with the visualized output (e.g., to specify regions of interest [60]). Due to the increasing availability of ubiquitous mobile devices such as smartphones and tablets, visualization systems also increasingly make use of the opportunities of (multi-)touch interaction [61]. Evaluations showed that these interfaces provide a quite natural, direct interaction within 3D virtual environments [62], and may outperform mouse input for certain tasks in terms of completion times [63]. However, touch user interfaces also adhere to certain challenges such as the intuitive mapping from 2D input to 3D manipulations [64, 65]. Our work provides an extensible interface for user interaction to parameterize image filtering on a spatial, thematic, and semantic basis of the visualized scene contents, e.g., via mouse, touch gestures, or textual input. The interaction framework integrated in our approach is practically used for exploration, navigation and analysis tasks performed with virtual 3D city models. In particular, we provide a blueprint for decoupling interaction from concrete visualization techniques using concepts of image-based rendering, *G-buffers* [45] and distance transforms for parameterization [66], and thus provide an extensible system design for software developers.

Finally, we use our interaction framework to directly parameterize our approach according to explicit and implicit view metrics (e.g., region interest, view distance). Our interaction framework explores interface schemes to allow users to attain both focused and contextual views of their information spaces. In particular, we use interactive lenses as established means to facilitate the exploration of complex 3D scenes [60], which are quite versatile in their parameterization. Previous approaches have been provided via the *magic lense* metaphor [67] and extended for 3D scenes [68]. The concept has also been explored for NPR [69] and 3D virtual environments (e.g., for geospatial tasks, such as navigation, landscaping, and urban city planning [70]) to reveal information that is hidden in high-dimensional data sets. We demonstrate how LoA texturing can be parameterized via the *magic lense* metaphor to provide context-based style variances for information highlighting and abstraction.

4. Method

An overview of our approach is shown in Figure 6. It is designed for generic application, can be seamlessly integrated into existing 3D rendering systems and pipelines, is extensible for arbitrary 2D image filters, and has no particular requirements regarding the consistency of the input geometry (e.g., triangulated 3D meshes vs. 3D point clouds). Additional attributes (e.g., semantics) and textures for appearance and thematic information can be processed for content-based filtering and multitexturing (e.g., using CityGML [71] for 3D geospatial models).

Our implementation performs filtering prior to texture mapping, for which decoupled deferred texturing is proposed (Section 4.1). Filtering is performed using GPGPU separately on each mipmap level for LoA texturing (Section 4.2). The implementation enables real-time performance for local image filters via per-fragment and progressive filtering using visibility information, together with a caching mechanism to ensure that image parts are only filtered once for a given configuration (Section 4.3). Progressive filtering is based on a computational budget that is applied per rendered frame to ensure a responsible system during navigation or interaction. Parameter sets of image filters can be defined per texture channel (e.g., diffuse vs. normal maps) and may be used to define layers with different levels of abstraction. These are blended according to view metrics or regions of interest for focus+context visualization. All geometry and texture buffers are represented as stencil-routed A-buffers [72] to support an order-independent image blending of filtering effects (e.g., blueprint rendering styles [73]). Optional post-processing may be performed in screen space and combined with the filtering results (Section 4.4) such as for depth of field effects.

The approach can be parameterized at multiple stages to give explicit control over the filtering process, as well as implicitly by view metrics (e.g., viewing distance). In addition, it integrates an interaction framework to parameterize the filtering effects more directly and concisely, e.g., using interaction techniques such as (multi-)touch gestures or natural language interface to automatically parameterize magic lenses or highlighting techniques. Here, the main idea is to decouple the interaction

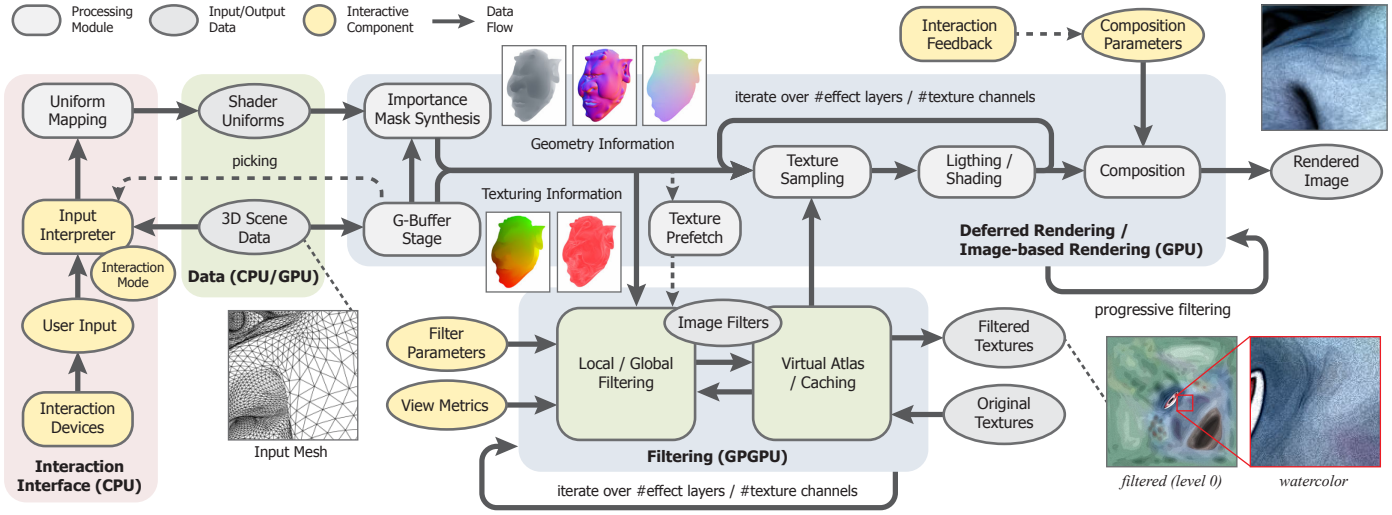

Figure 6: Schematic overview of our framework designed for interactive LoA texturing. The approach decouples the interaction interface and image filtering from rendering to provide an extensible architecture. Further, it performs the filtering prior to texture mapping to process texture data coherently with respect to linear perspective, and to preserve spatial relationships. (Jerry the Ogre © Keenan Crane. All rights reserved.)

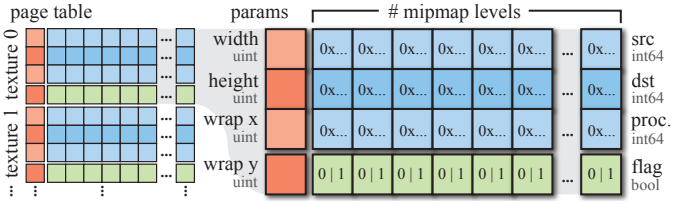

Figure 7: Layout of the virtual page table used for texturing of original and filtered texture data on the GPU.

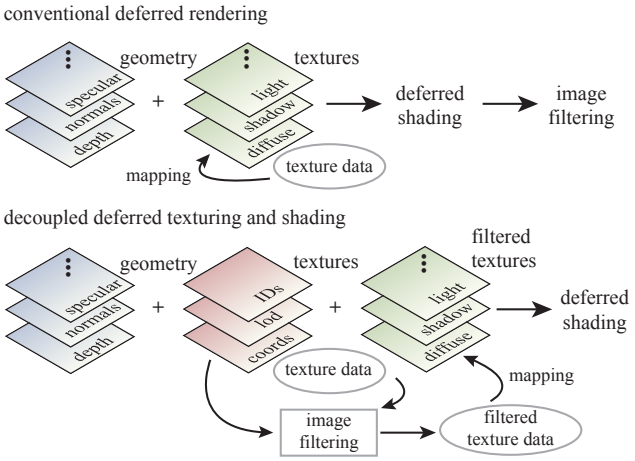

Figure 8: Conventional deferred rendering performs texture mapping in the geometry stage (top). By contrast, we propose decoupled deferred texturing for perspective-coherent filtering (bottom).

interface from rendering, and parameterize uniform buffers or projected distance maps as an abstraction layer in-between. Interaction devices and techniques are interpreted according to a pre-configured interaction mode, and mapped to a functional description of focus+context definitions. These definitions are evaluated in the deferred rendering stage to dynamically compute importance masks and perform image filtering according to a user-defined stylization preset.

In the following, we first focus on the technical aspects of the filtering processes that are performed on the GPU. Afterwards, the interaction framework used for parameterization is described in more detail (Section 5).

4.1. Decoupled Deferred Texturing

Our goal is to provide high-quality, interactive LoA visualization of general, textured 3D scenes by image filtering that complies with the key aspects named in Section 1. To this end, texture mapping is decoupled from the geometry pass and postponed by deferred texturing to transfer the filtering to texture space (Section 2).

First, in a pre-process stage, textures are assigned unique identifiers $T_{\mathrm{ID}} \in \mathbb{N}$. After computing the mipmap pyramids, a set of levels $(T_{m_0}, T_{m_1}, ..., T_{m_n})$ is defined per scene texture. The mipmap levels are transferred to GPU texture memory and referenced by their virtual address to enable *bindless texturing* with random read/write access. In addition, these virtual addresses are used for *virtual texturing* to enable a dynamic resolution of texture identifiers during processing. We use a page table with a memory layout shown in Figure 7. For each registered texture, this table references its dimensions, wrap modes required to handle filtering across texture borders (e.g., clamped vs. mirrored), and the virtual address of each mipmap level. The total number of mipmaps is adapted to the maximum texture dimension that can be processed by the GPU. Afterwards, rendering is performed in a series of three stages:

R1 ***Geometry Stage:*** Conventional deferred rendering stores all processed surface information in a *G-buffer* [45], including texture sampling to synthesize diffuse, normal and thematic maps used for visualization. By contrast, our implementation computes additional buffers related to texture information: identifiers, coordinates, and mipmap

LoD (Figure 8). Texture identifiers are assigned once in a pre-process stage and uniquely reference texture objects linked to the virtual address space.

R2 ***Filtering Stage:*** Texture parts required for rendering are sampled and filtered using the *G-buffer* information of the first stage. The results are written to separate texture buffers and used together with the original texture data as input for shading.

R3 ***Shading Stage:*** The results of the geometry and filtering stages are used for *deferred texture mapping*. At this stage, the texture information stored in the *G-buffer* is used to easily toggle between original and filtered texture variants. Optional filtering in screen space may be performed for post-processing effects, such as edge detection for contour enhancement or Gaussian filtering for depth of field.

The following sections describe the filtering and shading stages (R2/R3) in more detail.

4.2. Image Filtering

Once the *G-buffer* has been computed, the following three basic stages are performed for each render pass:

F1 ***G-buffer mapping:*** The *G-buffer* is mapped as an arrayed 2D graphics resource, where each pixel (x, y) is mapped to the texturing information $(T_{\text{ID}}, T_{\text{coord}}, T_{\text{lod}})$.

F2 ***Texture prefetch:*** Relevant textures required for rendering are determined. For each fragment in the *G-buffer*, the correspondent mipmap levels $(T_{m_i}, T_{m_{i+1}}) \leftarrow (T_{\text{ID}}, T_{\text{lod}})$ are computed in parallel and stored in a global structure using atomic operations. To avoid redundant filtering, a processing flag is set for each mipmap level in the virtual page table (Figure 7).

F3 ***Image filtering:*** Each unique T_{m_i} is filtered with the correspondent image filter and configuration. Additional borders may be introduced according to the wrap modes $(T_{\text{wrapx}}, T_{\text{wrapy}})$ to avoid visible seams when texturing. The results are written to the destination buffer and the processing flag is set.

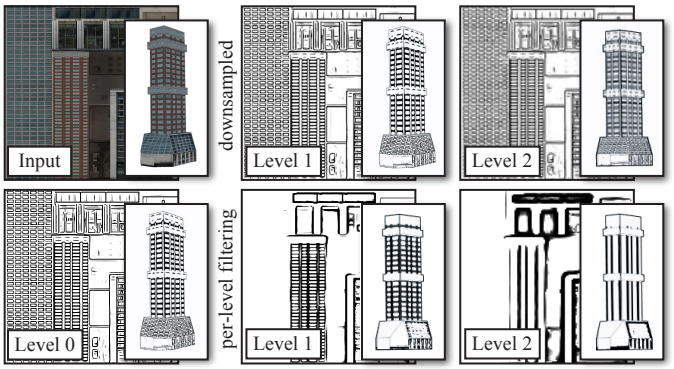

Figure 9: Example of LoA texturing by means of trilinear and DoG filtering: (top) downsampled filtered input, (bottom) input downsampled and then filtered per mipmap level.

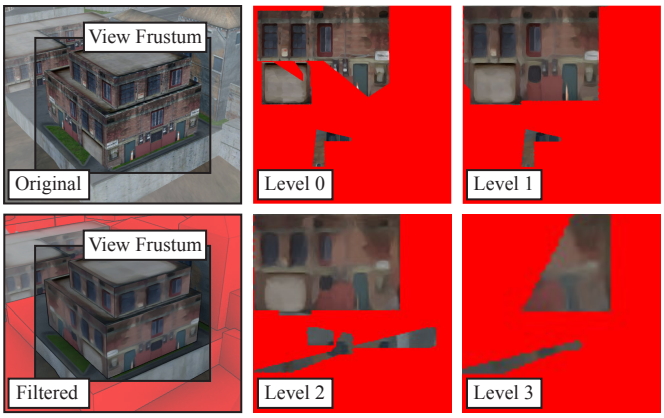

Figure 10: Local filtering is performed only for visible texture parts as performance optimization, whereas unused texture parts are uninitialized (red). Geometry outside the view frustum (bottom left) is only textured for reused parts.

Image filtering (F3) is performed separately on each mipmap level. Afterwards, the results are blended in the shading stage by trilinear filtering. Compared to filtering the highest mipmap level and downsampling the intermediate results, the approach enables a progressive LoA texturing (Figure 9). In this manner, visual clutter can be reduced significantly in areas of high perspective compression without requiring the adaption of filter parameters such as kernel sizes. Special care is required when mipmapped texture atlases are used for filtering to avoid bleeding across image tiles. Typically, this problem is addressed by introducing additional spaces between tiles, but it does not ultimately solve the problem for wide filter kernels or global operations. Virtual and bindless texturing alleviates this problem but requires using non-packed textures as input. An alternative approach may define tile masks per mipmap level for thresholding; however, this method increases the memory footprint.

4.3. Optimization Techniques and Enhancements

Dependent on the computational complexity of an image filter, the filtering stage (F3) could take significant processing time and stall the rendering stage. To this end, we introduce per-fragment filtering, caching, and progressive filtering for optimization.

Per-fragment Filtering. Because in most cases only small parts of textures are used for rendering and a lower texture LoD is selected in background regions of 3D perspective views, visibility-driven filtering allows for significant performance improvements, i.e., amortizing the computational overhead over a screen-space filtering. Therefore, per-fragment filtering is introduced by using the *G-buffer* to naturally process only the information required for rendering (Figure 10). This is achieved by integrating the *texture prefetch* in the filtering process, for which the texels required for trilinear filtering are determined for each fragment synthesized after rasterization. For trilinear filtering, this involves the four related texels used to perform bilinear filtering on each mipmap level (i.e., up to eight texels in total). The correspondent filtered values are then computed in parallel. Here, we used a top-down filtering approach, i.e., information required for filtering is recursively resolved on-demand during processing.

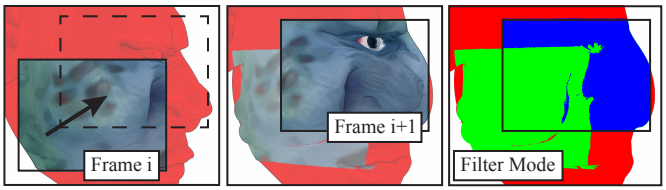

Figure 11: Filtered texture parts are cached (green), reused for subsequent frames, and combined with new filtered parts (blue).

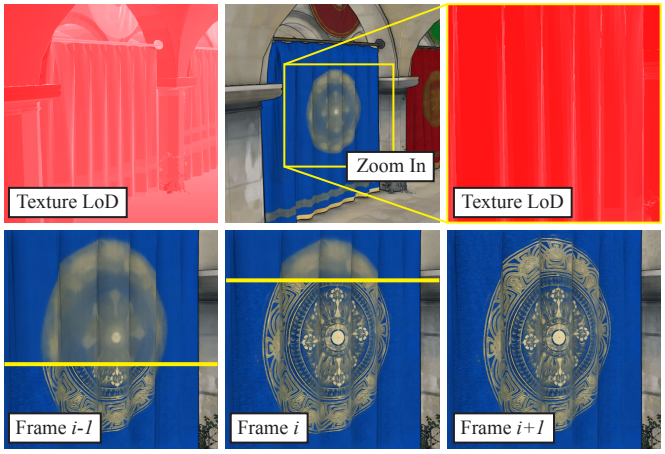

Figure 12: Example of a textured 3D scene that is progressively filtered using a computational filtering budget per render pass to maintain interactive frame rates. (Sponza Atrium scene © Marko Dabrovic and Frank Meinl from Crytek. All rights reserved.)

Caching. A caching mechanism is used so that texture parts are only filtered once for a given configuration and reused for subsequent render frames (Figure 11). The virtual page table (Figure 7) references additional image masks $(T_{p_0}, T_{p_1}, ..., T_{p_n})$ per mipmap level that indicate whether a pixel has already been processed.

Progressive Filtering. Some local and many global filters that solve an optimization problem do not perform at interactive frame rates. Filtering the highest mipmap levels first and using them as fallback is used for progressive filtering. Our system uses a computational budget that is applied per rendered frame and can be interactively configured. Detail information is then progressively blended in subsequent render frames (Figure 12). This procedure also ensures a responsive system during interaction, e.g., when adapting filter parameters at run-time. In addition, it prepares an easy deployment on multi-GPU systems to decouple computationally expensive global filters from rendering, but this remains subject to future work (Section 7).

The filtering kernel for both optimization techniques is summarized in Algorithm 1. Using these techniques, the computational cost of processing each texel over post-processing can be amortized, on the one hand, because pre-filtered texture information serves as input via mipmapping and, on the other hand, because the caching strategies avoid reprocessing. Our performance evaluation in Section 6.4 demonstrates that this enables local image filters to process textured 3D scenes at real-time frame rates.

Algorithm 1: *Per-fragment filter kernel for texture data*

```
1  function local_image_filtering: begin
       Input: G-buffer G, texture page table P with color lookup P_C and
              process flag lookup P_F, filtering budget B
2      k ← 0                          /* global number of texels filtered */
3      for pixels p ∈ G do in parallel
4          (ID, lod, u, v) ← G (p)                    /* sample G-buffer */
5          if ID = 0 then                                  /* early out */
6              return
7          (T_0, T_1) ← P (ID, ⌊ lod ⌋ and ⌈ lod ⌉)    /* mipmap LoDs */
8          forall the (T, u_S, v_S) of textureGather (T_0)
9          and        (T, u_S, v_S) of textureGather (T_1) do   /* 8 */
10             if P (T, u_S, v_S) not marked as processed then
               /* start of critical section */
11                 if k < B then                /* threshold budget */
12                     P_F (T, u_S, v_S) ← mark as processed
13                     P_C (T, u_S, v_S) ← filtering (T, u_S, v_S)
14                     k ← k + 1                 /* filtered color */
15                 else                  /* progressive filtering */
16                     P_C (T, u_S, v_S) ← lookup (ID, lod, u, v)
17                 end
               /* end of critical section                          */
18         end
19     end
20 end
```

Multitexturing and Context-based Filtering. To enable filtering of multitextured 3D scenes, the *G-buffer* is enhanced as follows (Figure 13). First, each fragment synthesized by the rasterization is assigned two sets of texture coordinates together with multiple texture identifiers defined per texture channel. Filter configurations can then be defined per texture channel to enable a content-based filtering that is adapted to 3D model semantics. Second, fragments are buffered in depth and a sorting is performed in a post-process stage to enable order-independent transparency effects. For this, the G-buffer is extended to a stencil-routed A-buffer [72] encoded as 3D texture for GPGPU processing. Depth sorting and image blending of the A-buffer is performed during shading. Third, different filters and configurations can be defined per input texture, for which the virtual texture table is extended by correspondent destination buffers. The filter results are then blended according to view metrics (e.g., viewing distance) or predefined regions of interest using image masks for focus+context visualization (e.g., stylized foci [4]). Examples that use these enhancements are presented in Section 6.

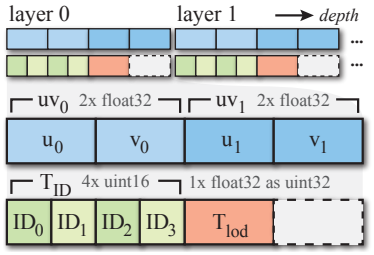

Figure 13: Extended *G-buffer* used for depth buffering and multitexturing.

The number of texture coordinates and channels is not ultimately defined but should be limited to bound memory consumption. Because all processes are designed for generic application, the *G-buffer* can be extended easily by additional attributes.

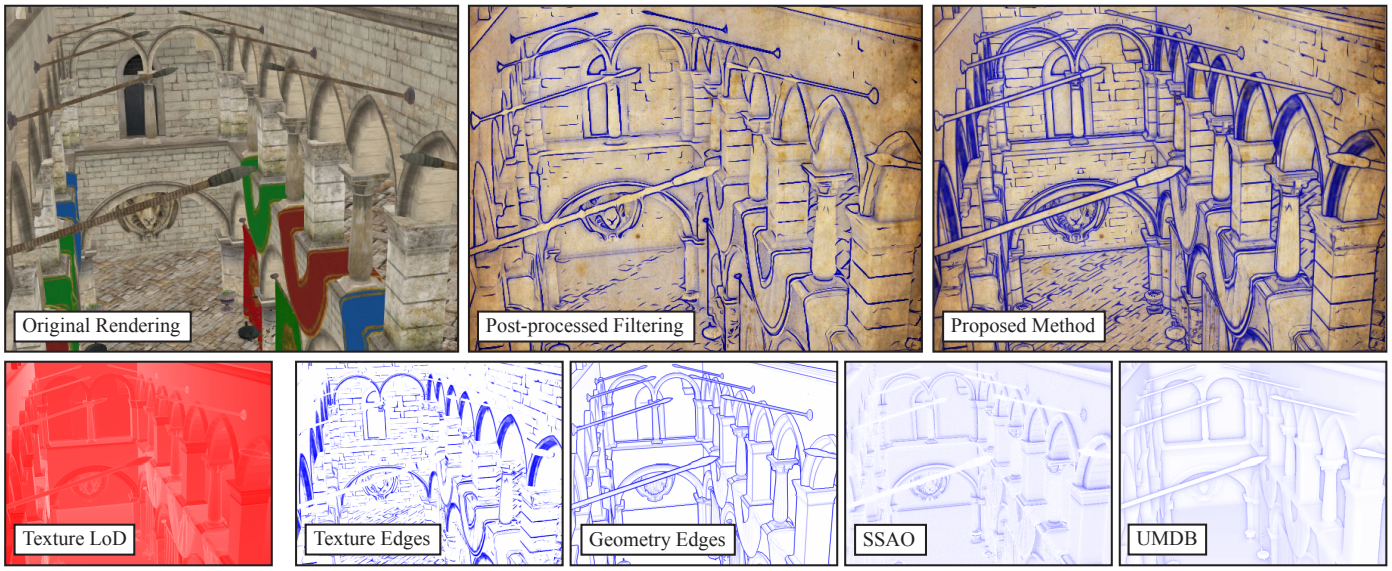

Figure 14: Example of post-processing effects used in our system. Edge detection is decoupled into DoG filtering of texture and geometry information. The results are combined with screen-space ambient occlusion (SSAO), unsharp masking the depth buffer (UMDB), and a background texture to compose the final images.

4.4. Deferred Shading and Composition

Once the *G-buffer* is computed and filtering is performed, the results are used as input for texture sampling, which is independently performed from shading and lighting. Optionally, screen-space ambient occlusion [74] and unsharp masking the depth buffer [75] can be performed using normal and depth information of the *G-buffer* to improve depth perception further. These effect layers can be individually amplified, colorized, and blended by regular image compositing [76]. Results that include DoG filtering are combined with an image-based edge enhancement technique [34] to include silhouette, border, and crease edges according to depth, normal, and object identifier information [45]. In contrast to traditional DoG filtering in screen space, our method is able to decouple edges based on texture and geometry information. Figure 14 and the accompanying video demonstrate that this approach produces much more accurate filtering results with respect to linear perspective. In addition, it provides real-time frame rates and frame-to-frame coherence without requiring specialized methods such as texture advection [15].

5. User Interaction

Our framework gives full control over the different parameters defined per image filter, including kernel size, quantization intensity, sensitivity of edge detection, adaptive smoothing in a post-process stage, and weights of flow fields (see Figure 15 for an excerpt). In addition, users are able to define parameters to control the composition of the filtering results:

- texture channel semantics with filters and configurations defined per channel for content-based LoA texturing;
- view metrics and region masks together with transition parameters used for blending layered filtering effects;

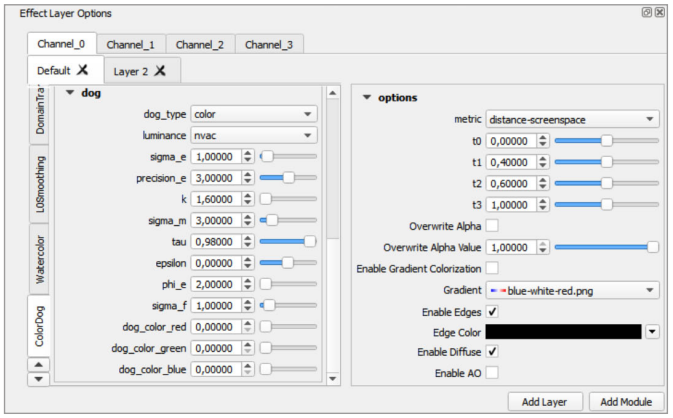

Figure 15: Excerpt of the graphical user interface of our framework for the parameterization of image filters (left) and effect layers (right), here exemplified for the FDoG filter [18].

- colors for enhanced geometry edges, screen-space ambient occlusion and unsharp masking effects;
- multi-sampling and order-independent transparency parameters used for image composition.

For the practical usage of our framework in real-world scenarios, in particular for non-expert users, however, adjusting these parameters can be cumbersome and such cases thus call for more high-level and direct interaction interfaces. For effective visualization of 3D scenes, direct user interaction is a critical design aspect to visualize as much information as needed for focus and as little as required for context. Four major interface schemes have been identified for focused and contextual views [77]: zooming, focus+context, overview+detail, and cue techniques. Using these schemes to parameterize LoA texturing in 3D virtual environments, however, remains a challenging task, because these environments are often inherently complex with

respect to appearance and thematic information. Here, our major goal is to provide an interaction framework that seamlessly integrates the proposed method for LoA texturing, is extensible for custom interaction devices and techniques, and considers the following challenges:

1. The filtering and rendering stages should be decoupled from concrete interaction interfaces, and be parameterized via high-level operations to facilitate an easy deployment of new interaction devices and techniques.
2. Interactive frame rates should be maintained to provide a responsive system to the user.
3. Mapping of direct interaction from 2D into 3D space should be handled, e.g., with respect to occlusions [64].

In the following, we define a generic workflow on how user interaction can be mapped to definitions of focus and context and their graphical representations via LoA texturing.

5.1. Focus+Context Definition and Interaction Techniques

A concrete challenge for the design of an interaction framework is to strive for consistency [78] while having the user in control of parameterizing the LoA texturing. Here, a key observation is that no constraints regarding the input device or technique should be made, i.e., to allow users to use the best direct interaction method for parameterizing our framework for a given task and environment (e.g., desktop vs. mobile devices). Our main idea is to decouple the functional descriptions of focus and context from the concrete interaction device and filter configurations, e.g., so that mouse/keyboard, touch-based, natural language, or implicit gaze-based interfaces can be used equally to define regions of interest. Technically, the interaction interpretation is formulated as mapping the user-defined input to a high-level functional description for focus and context definition. Interaction modes are required for disambiguation and to avoid redundant mappings, but should be made as concise as possible (e.g., using quasi-modes [79]). Exemplary mappings include object selections by textual lookup using natural language inputs or *point-and-click* metaphors. Similarly, circular regions of interest may be defined via *pinch-to-zoom* metaphors or sketching (e.g., for illustrative cutaway rendering [80]. Technically, high-level descriptions can be mapped from parameter space into model space using logical collections of 3D scene data as input, enriched with descriptive information that is stored as attributes (e.g., encoded with CityGML for geospatial models [71]).

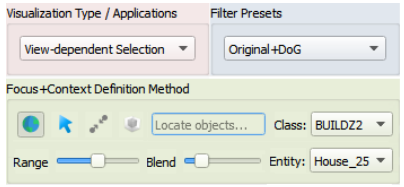

Figure 16: Overview of the interaction interface and tools implemented in our framework for parameterizing a focus+context visualization.

Dealing with 3D virtual environments, we distinguish between six types of focus definition. Figure 17 gives a conceptual overview of these types in a geospatial context:

1. *Object selection*: Highlighting single or groups of objects that are of major interest to a user.

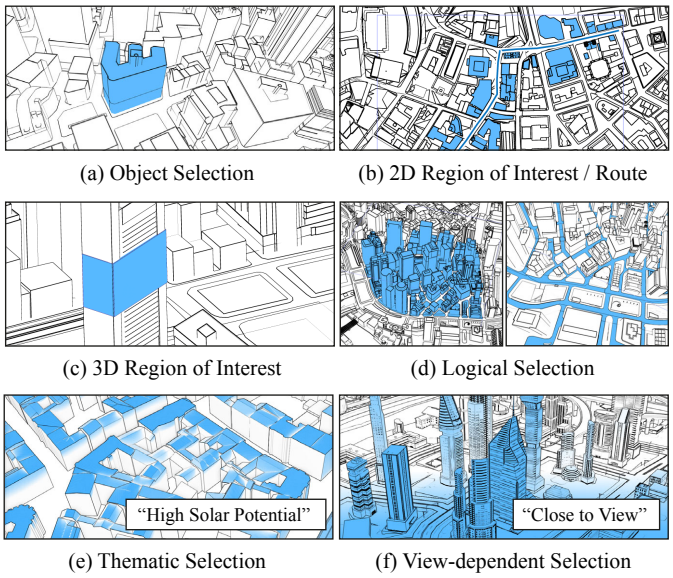

Figure 17: Focus definition types exemplified for virtual 3D city models.

2. *2D region of interest*: Highlighting objects that are located close to, or within a 2D region of interest.
3. *3D region of interest*: Spatial highlighting of objects or components with additional constraints in height.
4. *Logical selection*: Selection of objects or components with respect to semantic data, such as feature type (e.g., street networks in a routing scenario).
5. *Thematic selection*: Selection of objects or components with respect to thematic data, such as solar potential, and according to a range of interest.
6. *View-dependent selection*: Highlighting image regions according to view-based metrics (e.g., view distance or inclination).

Direct interaction for these types should trigger immediate visual feedback to symbolize a correspondent mode, for which we provide specialized shading effects. For instance, the boundaries of a circular region of interest are visualized using projected lines as visual cues.

Figure 16 gives an overview of the interaction interface implemented in our framework that resembles some of the focus+context definition types shown in Figure 17. Besides techniques for orbital, zoom and pan navigation, this interface is (1) context-aware according to a user-defined application (e.g., highlighting regions of interest), (2) allows users to select filter configurations from pre-defined presets, and (3) provides techniques for interactive focus+context definition. The latter includes regional definitions via the *pinch-to-zoom metaphor*, highlighting of objects or regions of interest via a search bar, value-range sliders for numerical constraints, and on-screen pointing techniques for direct object selection. In addition, multiple interaction techniques may be equivalently used for parameterization, e.g., using on-screen pointing or textual input to locate and highlight 3D scene objects.

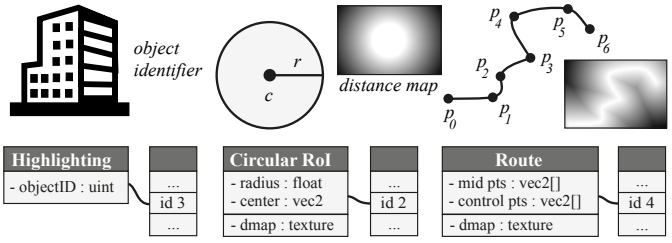

Figure 18: Focus definitions are mapped to uniform variables or are buffered as projected distance maps, which are evaluated during shading.

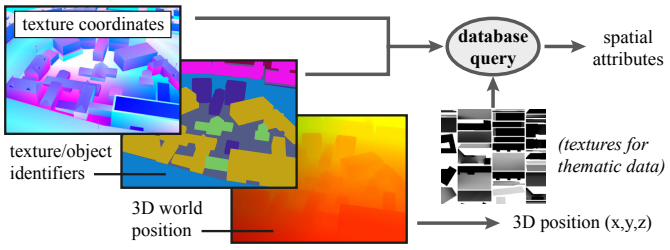

Figure 20: Texture coordinates, identifiers, and positions synthesized in the *G-buffer* are used to query additional attributes, stored in a database.

5.2. Shader Uniform Mapping

In our framework, focus and context definitions are either stored using GPU uniform buffers or mapped to distance maps (Figure 18), whose parameters are evaluated during shading. In the first case, these attributes are compared with the information synthesized in the *G-buffer* (e.g., object identifiers, 3D world position) for focus detection. In the second case, a Euclidean distance transform is performed to buffer line segments or 2D regions of interest. The synthesized distance maps are then projected onto the scene geometry and evaluated during shading. This approach enables image-based operations to be efficiently implemented, such as fragment-based thresholding of the Euclidean distance between objects (e.g., distance to a route, Figure 18).

In most interaction modes, a basic functionality is to map 2D inputs to 3D attributes via raycasting (e.g., object selection via touch interfaces [59]). It is typically performed using intersection tests with the 3D scene geometry, but is often too complex to be interactively performed. Instead, we use an image-based approach using the synthesized *G-buffer* information to query 3D scene attributes for the visible scene geometry ("picking"). Here, we observed that the synthesized world position, texture coordinates, and identifier information (i.e., objects, textures) are sufficient to query arbitrary attributes stored in a database (Figure 20). For instance, texture identifiers and coordinates directly map into texture space for a fragment-based information lookup, whereas object identifiers can be mapped to object-specific attributes.

5.3. Importance Mask Synthesis and Shading

The shader uniforms are evaluated using fragment shaders. For each definition type, a normalized importance mask is synthesized that indicates whether a fragment should be shaded for focus or context (Figure 19). Blend functions are utilized for image composition [76] and to enable smooth transitions between focus and context regions (Figure 21), but may also be configured for hard transitions, e.g., to avoid distorted color tones when using heterogeneous image filters. Finally, the importance masks are blended to enable multivariate effects (e.g., a route with a circular region of interest at the destination).

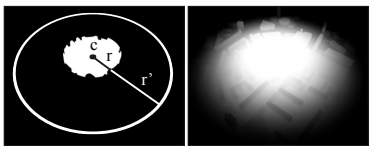

Figure 19: Example of a synthesized importance mask for a circular region of interest.

For view-dependent visualization, view or user-defined metrics are evaluated during shading. Here, we supply presets of image filter configurations to enable an automated LoA texturing setup. For instance, a blueprint rendering may be automatically selected to represent construction sites in an urban planning scenario. Post-processing of the synthesized importance masks with a screen-space distance transform may also be used for highlighting (e.g., glow effects [58]). Results using these interactive techniques for LoA texturing are presented in the next section.

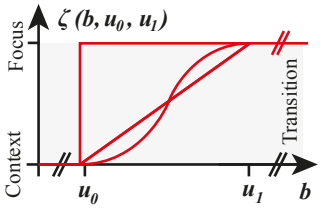

Figure 21: Transition functions used to blend focus and context regions.

6. Results and Discussion

This section presents implementation details, applications, evaluations, and limitations of our work.

6.1. Implementation

We have implemented our framework using C++, OpenGL, and GLSL, and the image filters on the GPU with CUDA. OpenSceneGraph was used as the rendering engine to handle 3D data sets. For global image processing, CUDA's *fast Fourier transform (cuFFT)* was used to implement the L_0 gradient minimization [23], the Deriche method [81] to perform Gaussian smoothing in constant time per input pixel, and recursive methods proposed by Nehab et al. [82] for box filtering. All image filters operate in RGB color space or CIE-Lab. Our implementation uses texture and surface objects introduced in CUDA 5 for virtual texturing along with the *OpenGL Interop* functionality for random read and write access. We assume that similar results should be achievable using OpenCL or compute shaders.

For bindless texturing, we used the OpenGL 4.4 extension `GL_ARB_bindless_texture`. The *G-buffer* is packed to RGBA or RGB texture channels and encoded as a stencil-routed A-buffer [72] for order-independent image blending. The parallel-banding algorithm [83] was used to perform work-load efficient distance transforms and compute importance masks for regions of interest. All results were rendered on an Intel® Xeon™ 4× 3.06 GHz with 6 GByte RAM and NVidia® GTX 760 GPU with 4 GByte VRAM. In addition, user interaction was tested with a 23.6" Lenovo® L2461xwa multitouch monitor.

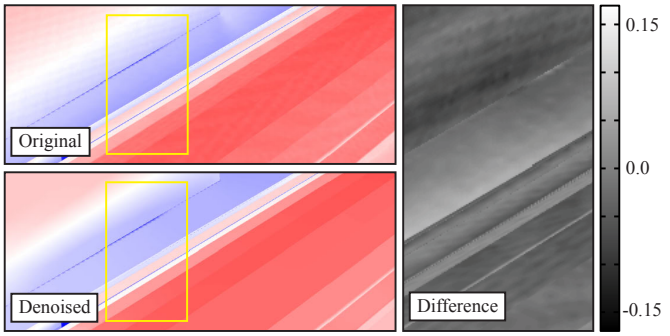

Figure 22: Edge-preserving denoising of texture-encoded thematic data via L_0 gradient minimization ($\lambda = 0.02, \kappa = 2$), projected onto surface geometry.

6.2. Applications

The individual applications of the respective image filters—e.g., HDR tone mapping, detail exaggeration, edge adjustment, or colorization—can be maintained because they are merely transferred to the texture domain. For instance, Figure 22 shows a result of our method using L_0 gradient minimization [23] for perspective-coherent denoising of texture-encoded thematic data and overall contrast enhancement. Here, solar potential was computed by a radiation summed up over a whole year.

Focus+Context Visualization. Highlighting regions of interest while removing detail in context regions is a major goal in the effective information transfer and for directing a viewer's gaze to important or prioritized information [3, 28]. Figure 25 shows a result of our framework to implement a *stylized focus pull* effect [4] based on the view distance. Emphasis is drawn to the respective image regions using the FDoG filter coupled with image-based enhancements of geometry edges for context regions. Our method is able to preserve the overall structure of objects in the background and in the context regions (e.g., windows on the building façades to emphasize spatial relationships). Because the image composition is performed in the deferred shading stage and multiple layer effects may be defined, our approach is generic with respect to filter combinations and extensible for further view metrics (e.g., region masks or view angles). Here, we coupled the *pinch-to-zoom* metaphor with our interaction framework to directly change the graphical representation in regions of interest. Based on a multitouch interface, the user starts pointing with two fingers and spans—via the zoom metaphor—the range of interest in projected world space. A user-defined preset for image filtering then automatically adjusts the LoA texturing in the focus region, providing optional smooth transitions controlled via a blend slider. Alternatively, a natural language interface may be used to locate and highlight regions of interest via a search bar. Figure 24 exemplifies a semantic lens that automatically blends a detailed 3D model for focus with a map for context. Here, the *apparent greyscale* algorithm by Smith et al. [84] was used for filtering the context regions.

Thematic Visualization. The multitexturing support was used to enhance depth cues important for visualizing color-encoded thematic data in virtual 3D scenes. Figure 26 shows a parameterization in which the FDoG filter was used to detect edges in the color textures and blend the result with the thematic information. Compared to an edge detection that only processes geometry information, our approach also enhances structural information that is not explicitly modeled as geometry, but is captured by aerial or terrestrial imaging. This way, the correlation between thematic data and surface structures is much more plausible. We explored approaches to directly parameterize a thematic visualization for information highlighting. Here, the implemented slider interface for value-range thresholding was used to parameterize the range of interest for thematic data. Figure 27 shows a result, where areas with a high solar potential are visualized with detailed graphics, and areas with a low solar potential are automatically filtered, providing smooth transitions in-between.

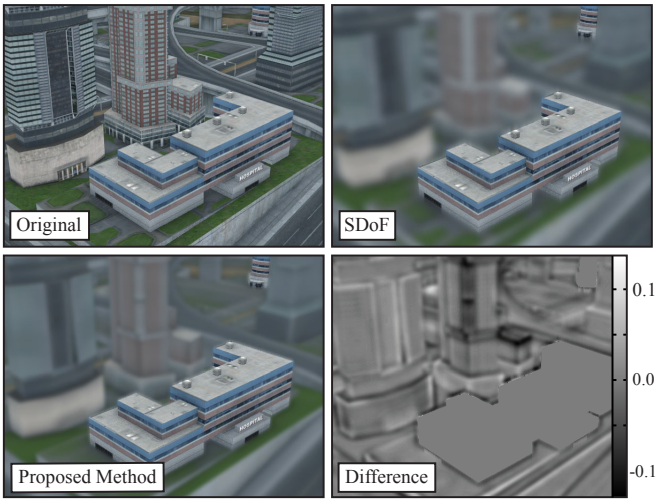

Figure 23: Gaussian smoothing for semantic depth of field. We used our LoA texturing method for perspective-coherent Gaussian smoothing of color maps ($\sigma_t = 3.5$) prior to additional smoothing in screen space ($\sigma_s = 3.5$). Compared to the traditional approach, which only filters in screen space ($\sigma_s = 7.5$), our method preserves scene structures in context regions better at a similar LoA.

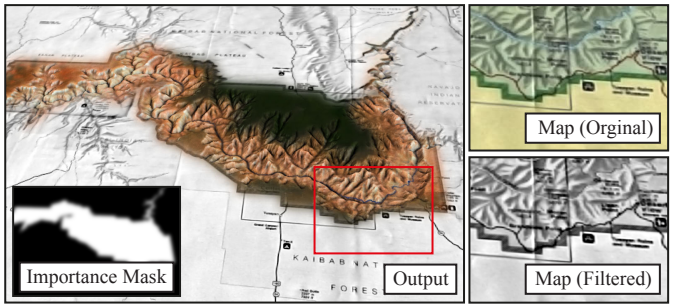

Figure 24: Semantic highlighting (*Grand Canyon National Park*) using a 3D terrain model for focus and a 2D map for context. The map is filtered using the *apparent greyscale* algorithm by Smith et al. [84] ($p = 0.5, k = \{0.5, 0.5, 0.5, 0.5\}$).

Semantic Depth of Field. Depth of field is known to direct the pre-attentive cognition of prioritized information. Using our method, we have implemented a variant of the *semantic depth of field* (SDoF) effect [57], in which scene objects are selectively highlighted via image masks to direct a Gaussian smoothing in screen space. First, diffuse maps are filtered in texture space, i.e., to control the LoA of textured surfaces. After-

Figure 25: Example of a stylized focus pull where the distance-based focal plane is gradually directed from the foreground to the background. A watercolor and DoG filter were used to draw emphasis to regions of interest and preserve depth cues in context regions (e.g., building windows).

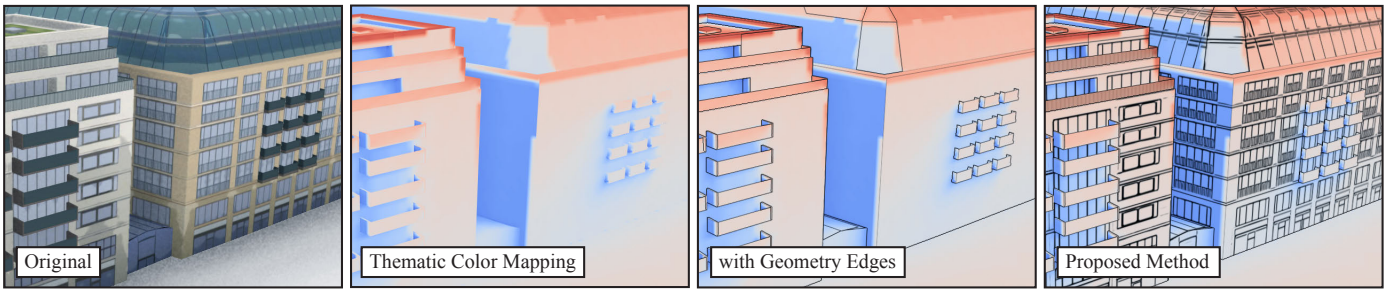

Figure 26: An example for thematic data visualization, where our framework was used to combine geometry edges and DoG filtered color textures and, hence, improve the visualization of color-encoded thematic data (here: solar potential) significantly.

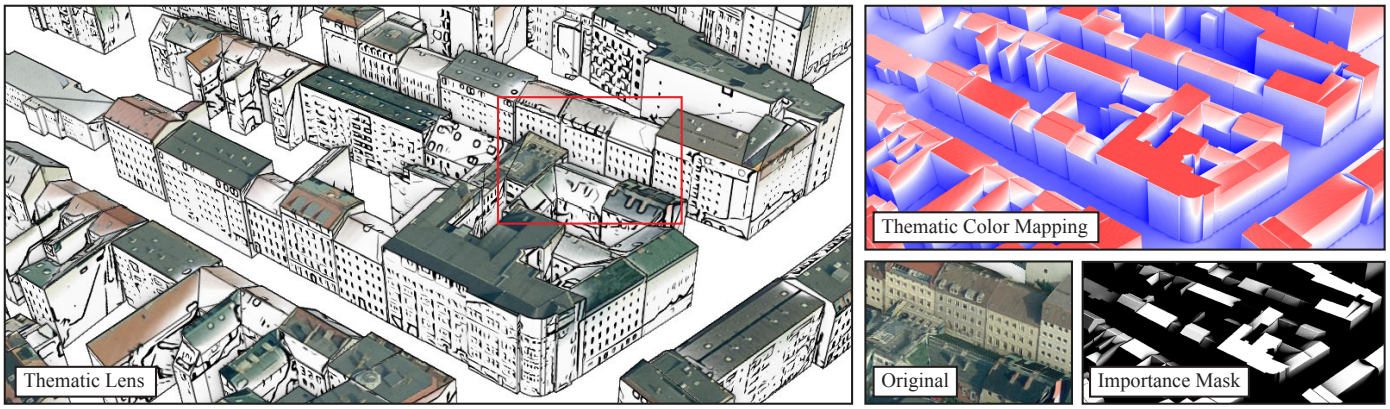

Figure 27: Interactive selection of "high solar potential data" using our framework to direct the filtering process to context regions. The user is able to control the focus and context definition via a value-range slider together with a blend slider to provide smooth transitions between both definitions.

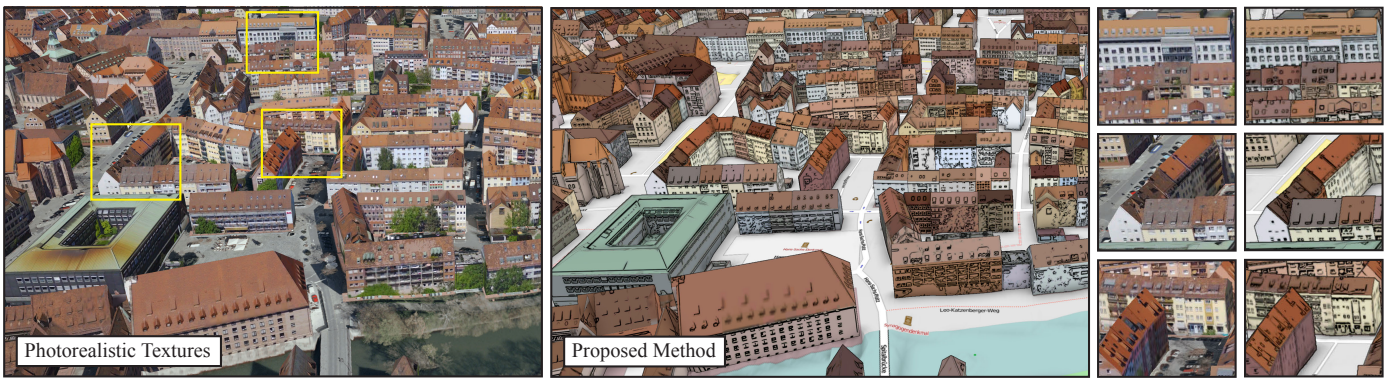

Figure 28: Cartography-oriented design of 3D geospatial information visualization for a virtual 3D city model (Nuremberg, Germany). Building façades (encoded with color textures) are filtered and colorized by their dominant colors per component (exterior walls vs. roofs), and blended with feature contours.

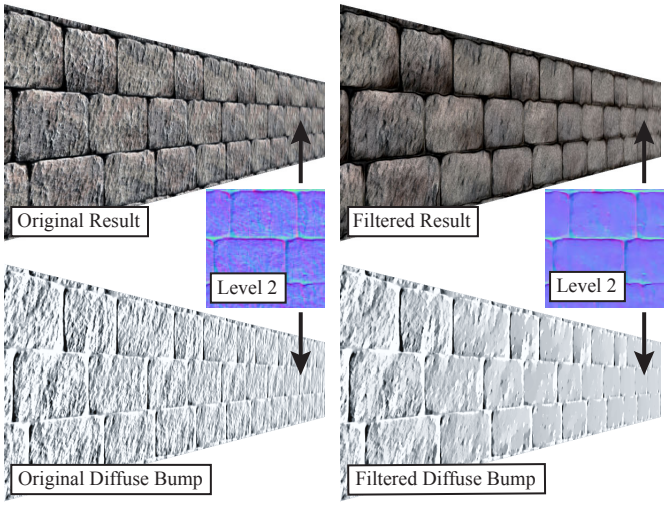

Figure 29: Smoothed bump mapping using a watercolor filter and domain transform (recursive mode, $\sigma_s = 30.0, \sigma_r = 0.25, N = 3$) for color and normal maps.

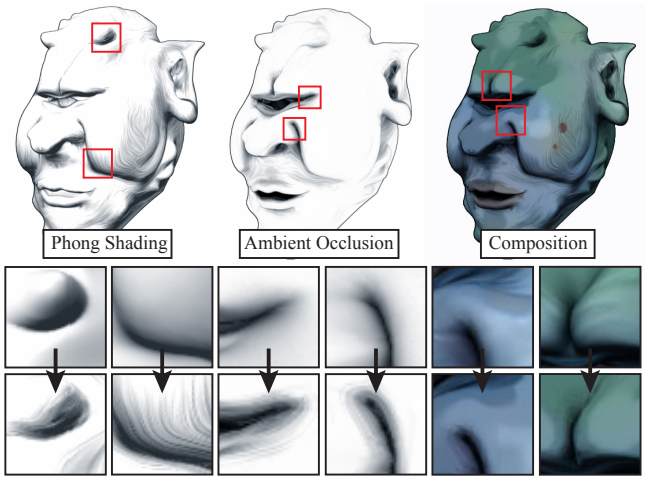

Figure 31: Stylization of Phong shading, ambient occlusion, and color maps. An oil paint filter was used for the normal and ambient occlusion maps, together with the abstraction filter by Winnemöller et al. [22] for color maps.

wards, regular Gaussian smoothing is performed in screen space to blur geometry edges. As shown in Figure 23, using this "dual" filtering approach enabled a clearer visualization of structures induced by geometry edges. By contrast, the regular screen-space approach requires wider filter kernels to achieve a similar LoA effect but at the cost of a considerably high degradation of scene structures. Our search interface was used to automatically parameterize the SDoF effect for object highlighting.

Cartography-oriented Design. Graphical core variables in the cartographic design of 3D building and site models often involve reduced color palettes, e.g., distinct colors for roofs and façades, combined with contour lines and strokes indicating features such as windows (Figure 30). We used our method to extract structural elements from photorealistic color textures together with an algorithm that automatically extracts dominant colors from regions with constant tone, which are weighted and aggregated via entropy-based metrics. Figure 28 shows a result of our approach applied to a virtual 3D city model. Compared to typical photorealistic depictions (e.g., as provided by 3D geovirtual environments such as Google Earth), our approach improves feature contrasts, provides a decluttered representation with inherent shadow removal, and expresses uncertainty. We believe that this kind of visualization can be of major interest for routing applications, where often only a few selected information are required for orientation.

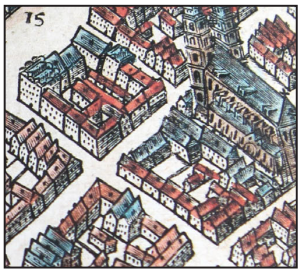

Figure 30: Design of buildings and sites in a historic, hand-drawn map.

Geometric Detail Removal. Bump or displacement mapping is an essential process for enriching shading and lighting effects by geometric detail. We used our method to perform edge-preserving smoothing of normal maps to coarse bump mapping with respect to the linear perspective. Figure 29 shows the results of a domain transform [38] applied to normal maps, where mipmapping enables a smooth transition between the different levels of structural abstraction.

Stylized Shading. We experimented with filtering normal and ambient occlusion maps to achieve stylized shading and lighting effects. The rich parameterization options of our framework give artists creative control over this process. For instance, Figure 31 shows how an oil paint filter—based on a smoothed structure tensor—was used to apply Phong shading and ambient occlusion with a sketchy style. Similar directions were proposed by DeCoro et al. [85] for stylized shadows and toon shading by Barla et al. [53] for general LoA; however, without the capability for flow-based image abstraction. By contrast, our approach provides stylized variants of texture maps that includes salient structures, which are blended by conventional shading. Hence, it is especially useful to interactively stylize photorealistic texture maps (e.g., captured by terrestrial photography).

Blueprint Rendering. Finally, the capability to layer filtering in depth was observed by a novel blueprint rendering approach. In contrast to image-based techniques that solely operate on *G-buffer* information [73], we also used a DoG filter for perspective-coherent abstraction of diffuse maps that preserves texture gradients (Figure 32 left). The filtering results were colorized and blended by order-independent transparency. Using our interaction framework, interiors of 3D models model can be made interactively explorable via the magic lens metaphor [67, 68]. Here, a user is able to span regions of interest via direct touch interaction and shift the lens to a desired location. The focus area may then be visualized with our blueprint rendering approach (Figure 32 right). The accompanying video demonstrates an interactive parameterization for a multitouch display. In addition, it demonstrates frame-to-frame coherence for all of our results. Here, the coherence primarily comes from processing the respective mipmap levels according to perspective projection, *prior* to trilinearly filtering the results on the GPU for smooth interpolation (Equation 2).

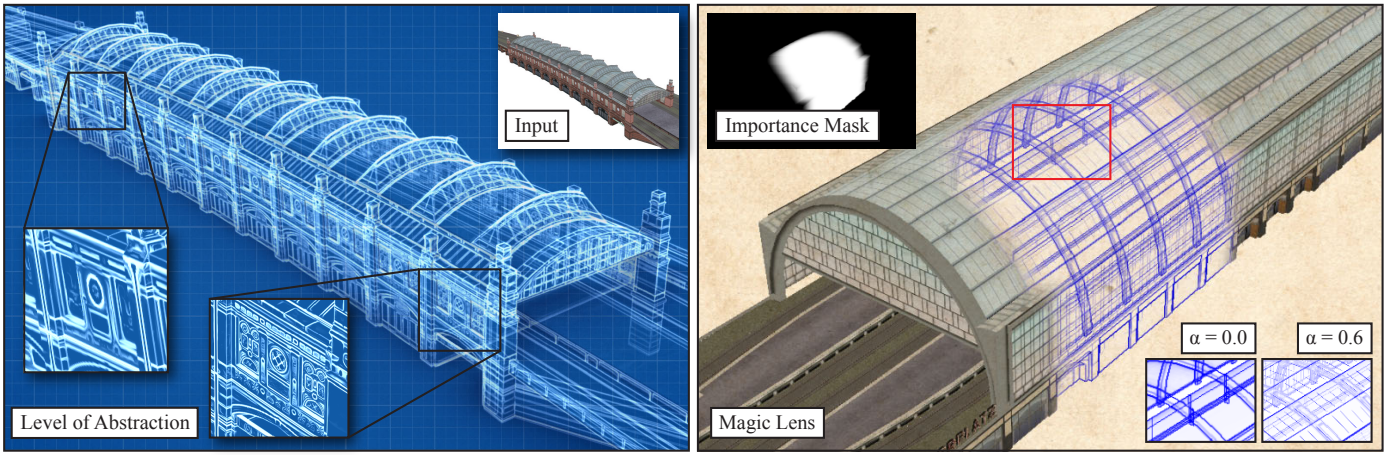

Figure 32: Blueprint rendering based on geometry edges and FDoG filtered color maps. The filtering is performed on each depth layer, the results are then sorted in depth and blended in a post-process stage. Left: LoA capabilities of our approach, where the close-ups represent the first (visible) depth layer. Right: A magic lens interactively parameterized via a touch interface to direct a blueprint rendering to a region of interest, highlighting interior structures.

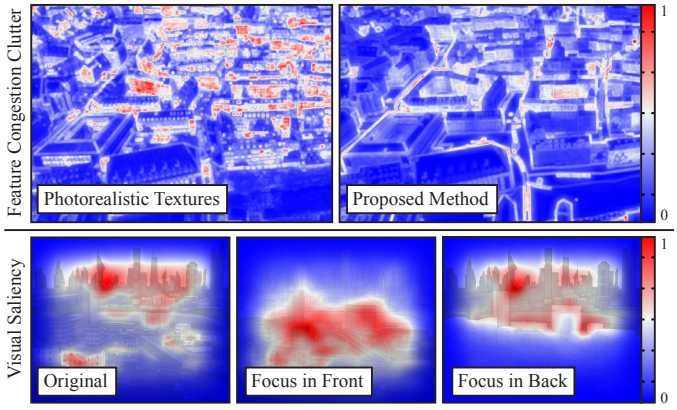

Figure 33: Visual clutter analysis (top) for Figure 28 and visual saliency analysis (bottom) for Figure 25, each compared to the respective photorealistic version, using the algorithms by Rosenholtz et al. [86] and Harel et al. [87].

6.3. Visual Clutter and Saliency Evaluation

To demonstrate the benefits of LoA texturing for an effective information transfer, we compared visual clutter and saliency maps of our outputs with the original, high-detail renderings. Using the *feature congestion measure* proposed by Rosenholtz et al. [86], Figure 33 demonstrates how our cartography-oriented visualization approach reduces visual clutter, thus being able to reliably draw attention to important features of the virtual 3D scene. In addition, Figure 33 demonstrates how the visual saliency of our the stylized focus pull effect yields high saliency within the respective focus regions. Similar results have been qualitatively verified using an eye tracker [3, 4].

6.4. Performance Evaluation

Because our framework was designed for general-purpose application, its performance greatly depends on which image filters are used—which we identified as bottleneck—and how many effect layers are defined. We rendered the virtual 3D city scene depicted in Figure 25 with the multi-scale anisotropic Kuwahara filter ($N = 4$) [26] and the proposed SDoF variant with the system setup described in Section 6.1. The scene is composed of 540 unique 3D objects with 15 texture atlases (each 1024×1024 pixels). We defined a fly-through sequence that lasts 15 seconds with filter caches cleared prior to each iteration. The results provided in Table 1 show that the performance scales with the screen resolution, reaching interactive frame rates in HD resolutions and real-time performance when using our caching concept. Notice how per-fragment filtering without caching is almost on par with conventional post-process filtering in HD screen resolutions, thus it could also be used for dynamic textures. The timings include all filtering and rendering cycles with potential memory transfers. The memory consumption is proportional to the screen resolution with respect to the *G-buffer* and proportional to the number of filter layers with respect to cached textures targets. We believe that the consumption can be significantly decreased when using sparse textures (OpenGL 4.3).

Table 1: Performance evaluation of our framework using the multi-scale Anisotropic Kuwahara filter (AKF) [26] and the SDoF configuration used to produce Figure 23 (average and minimum frames-per-second). Setups: (1) filtering entire mipmap levels with caching enabled, (2) per-fragment filtering with caching disabled and (3) enabled, (4) filtering in a post-process stage.

	Screen Res.	Setup 1	Setup 2	Setup 3	Setup 4
AKF	800 × 600	75.6 (12)	12.9 (9)	87.9 (44)	17.2 (16)
	1280 × 720	51.9 (12)	11.5 (10)	67.3 (26)	14.7 (14)
	1920 × 1080	31.5 (11)	11.3 (10)	37.3 (14)	11.8 (11)
SDoF	800 × 600	53.8 (23)	12.3 (8)	66.3 (54)	29.4 (28)
	1280 × 720	44.1 (20)	11.1 (8)	52.9 (41)	19.6 (18)
	1920 × 1080	30.2 (13)	8.6 (7)	33.5 (24)	10.0 (9)

6.5. Limitations

In contrast to image-based artistic stylization, our methods are not able to inherently control the LoA of the scene geometry. However, this enables a more controlled geometric abstraction by specialized techniques. Alternatively, our methods may be combined with filtering in a post-process stage to selectively

filter across object boundaries. In addition, UV mapped textures are required as input, otherwise a spatial filtering (e.g., for the SDoF variant) is not practical. There is also room for improvement of our system's performance. As a current limitation, interactive filtering cannot be maintained when solving sparse linear systems (e.g., for image decomposition) because local image filtering cannot be performed. This also applies to iterative approaches that are resistant to parallelization. Outsourcing computationally expensive filters to multi-GPU systems could alleviate this problem, which is supported by progressive and decoupled filtering but it remains subject to future work.

7. Conclusions and Future Work

In this paper, we have presented interactive image filtering for LoA texturing of virtual 3D scenes. Our decoupled deferred texturing concept enables to process texture maps coherently with respect to linear perspective by arbitrary image filters to preserves depth cues. We fashioned a caching concept and optimization schemes by per-fragment and progressive filtering that enable interactive or real-time frame rates. In addition, we proposed a framework that is extensible by custom image filters and interaction techniques to parameterize the filtering process via mouse, touch, or natural language. Several results demonstrate its manifold application to the fields of visualization.

We see multiple directions for future work. First, sketch-based interfaces could be coupled with our interaction framework to provide an authoring tool for aesthetic renditions of virtual 3D scenes. Second, we plan to conduct a qualitative user study next to our quantitative evaluation to confirm that our approach is able to significantly improve visualization tasks. Third, our applications demonstrate that decoupled deferred texturing and shading is able to effectively map user interaction to focus and context definitions. However, the integration of new interaction techniques still requires efforts to map the user input to GPU resources (e.g., uniform buffers). Here, an extensible rendering backend remains to be explored.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable comments. The Berlin 3D city model has been provided by Berlin Partner for Business and Technology and the Business Location Center, Berlin. The Nuremberg 3D city model is used by courtesy of the City of Nuremberg, Bavarian Agency for Surveying and Geoinformation (2013). We would also like to thank our technology partner 3D Content Logistics, Marko Dabrovic and Frank Meinl from Crytek for the Sponza scene used in Figure 12 and Figure 14, and Keenan Crane for Jerry the Ogre used in Figure 6, Figure 11 and Figure 31. This work was funded by the Federal Ministry of Education and Research (BMBF), Germany, within the InnoProfile Transfer research group "4D*n*D-Vis" (www.4dndvis.de).

References

[1] Haeberli, P., Segal, M.. Texture Mapping as a Fundamental Drawing Primitive. In: Eurographics Workshop on Rendering. The Eurographics Association; 1993, p. 259–266.

[2] Ware, C.. Information Visualization: Perception for Design. San Francisco: Morgan Kaufmann Publishers Inc.; 2004.

[3] Santella, A., DeCarlo, D.. Visual Interest and NPR: an Evaluation and Manifesto. In: Proc. NPAR. ACM; 2004, p. 71–78. doi:10.1145/987657.987669.

[4] Cole, F., DeCarlo, D., Finkelstein, A., Kin, K., Morley, K., Santella, A.. Directing Gaze in 3D Models with Stylized Focus. In: Proc. EGSR. The Eurographics Association; 2006, p. 377–387. doi:10.2312/EGWR/EGSR06/377-387.

[5] Winnemöller, H., Feng, D., Gooch, B., Suzuki, S.. Using NPR to Evaluate Perceptual Shape Cues in Dynamic Environments. In: Proc. NPAR. ACM; 2007, p. 85–92. doi:10.1145/1274871.1274885.

[6] Redmond, N., Dingliana, J.. Investigating the Effect of Real-time Stylisation Techniques on User Task Performance. In: Proc. APGV. ACM; 2009, p. 121–124. doi:10.1145/1620993.1621017.

[7] Tomasi, C., Manduchi, R.. Bilateral Filtering for Gray and Color Images. In: Proc. ICCV. IEEE; 1998, p. 839–846. doi:10.1109/ICCV.1998.710815.

[8] Gooch, B., Reinhard, E., Gooch, A.. Human Facial Illustrations: Creation and Psychophysical Evaluation. ACM Trans Graph 2004;23(1):27–44. doi:10.1145/966131.966133.

[9] Kyprianidis, J.E., Collomosse, J., Wang, T., Isenberg, T.. State of the Art: A Taxonomy of Artistic Stylization Techniques for Images and Video. IEEE Trans Vis Comput Graphics 2013;19(5):866–885. doi:10.1109/TVCG.2012.160.

[10] Pfautz, J.D.. Depth Perception in Computer Graphics. Ph.D. thesis; University of Cambridge; 2000.

[11] Goldstein, E.B.. Sensation and Perception. Wadsworth Publishing Company; 2010.

[12] Semmo, A., Trapp, M., Kyprianidis, J.E., Döllner, J.. Interactive Visualization of Generalized Virtual 3D City Models using Level-of-Abstraction Transitions. Comput Graph Forum 2012;31(3):885–894. doi:10.1111/j.1467-8659.2012.03081.x.

[13] Wanger, L., Ferwerda, J., Greenberg, D.. Perceiving Spatial Relationships in Computer-Generated Images. IEEE Comput Graph Appl 1992;12(3):44–58. doi:10.1109/38.135913.

[14] Surdick, R.T., Davis, E.T., King, R.A., Corso, G.M., Shapiro, A., Hodges, L., et al. Relevant Cues for the Visual Perception of Depth: Is Where You See it Where it is? In: Proc. Hum. Fact. Ergon. Soc. Annu. Meet.; vol. 38. SAGE Publications; 1994, p. 1305–1309. doi:10.1177/154193129403801912.

[15] Bénard, P., Bousseau, A., Thollot, J.. State-of-the-Art Report on Temporal Coherence for Stylized Animations. Comput Graph Forum 2011;30(8):2367–2386. doi:10.1111/j.1467-8659.2011.02075.x.

[16] Gooch, A.A., Long, J., Ji, L., Estey, A., Gooch, B.S.. Viewing Progress in Non-photorealistic Rendering through Heinlein's Lens. In: Proc. NPAR. ACM; 2010, p. 165–171. doi:10.1145/1809939.1809959.

[17] Semmo, A., Döllner, J.. Image Filtering for Interactive Level-of-Abstraction Visualization of 3D Scenes. In: Proc. CAe. ACM; 2014, p. 5–14. doi:10.1145/2630099.2630101.

[18] Kyprianidis, J.E., Döllner, J.. Image Abstraction by Structure Adaptive Filtering. In: Proc. EG UK TPCG. The Eurographics Association; 2008, p. 51–58. doi:10.2312/LocalChapterEvents/TPCG/TPCG08/051-058.

[19] Williams, L.. Pyramidal Parametrics. ACM SIGGRAPH Comput Graph 1983;17(3):1–11. doi:10.1145/800059.801126.

[20] Donnelly, W., Lauritzen, A.. Variance Shadow Maps. In: Proc. ACM I3D. ACM; 2006, p. 161–165. doi:10.1145/1111411.1111440.

[21] Kyprianidis, J.E., Kang, H.. Image and Video Abstraction by Coherence-Enhancing Filtering. Comput Graph Forum 2011;30(2):593–602. doi:10.1111/j.1467-8659.2011.01882.x.

[22] Winnemöller, H., Olsen, S.C., Gooch, B.. Real-Time Video Abstraction. ACM Trans Graph 2006;25(3):1221–1226. doi:10.1145/1141911.1142018.

[23] Xu, L., Lu, C., Xu, Y., Jia, J.. Image Smoothing via L_0 Gradient Minimization. ACM Trans Graph 2011;30(6):174:1–174:12. doi:10.1145/2070781.2024208.

[24] Weickert, J.. Anisotropic Diffusion in Image Processing. Teubner Stuttgart; 1998.

[25] Kang, H., Lee, S., Chui, C.K.. Flow-Based Image Abstraction. IEEE Trans Vis Comput Graphics 2009;15(1):62–76. doi:10.1109/TVCG.2008.81.

[26] Kyprianidis, J.E.. Image and Video Abstraction by Multi-scale Anisotropic Kuwahara Filtering. In: Proc. NPAR. ACM; 2011, p. 55–64. doi:10.1145/2024676.2024686.

[27] Comaniciu, D., Meer, P., Member, S.. Mean Shift: A Robust Approach Toward Feature Space Analysis. IEEE Trans Pattern Anal Mach Intell 2002;24:603–619. doi:10.1109/34.1000236.

[28] DeCarlo, D., Santella, A.. Stylization and Abstraction of Photographs. ACM Trans Graph 2002;21(3):769–776. doi:10.1145/566654.566650.

[29] Bousseau, A., Kaplan, M., Thollot, J., Sillion, F.X.. Interactive Watercolor Rendering with Temporal Coherence and Abstraction. In: Proc. NPAR. ACM; 2006, p. 141–149. doi:10.1145/1124728.1124751.

[30] Criminisi, A., Sharp, T., Rother, C., Pérez, P.. Geodesic Image and Video Editing. ACM Trans Graph 2010;29(5):134:1–134:15. doi:10.1145/1857907.1857910.

[31] Mould, D.. Texture-preserving Abstraction. In: Proc. NPAR. The Eurographics Association; 2012, p. 75–82. doi:10.2312/PE/NPAR/NPAR12/075-082.

[32] Kang, H., Lee, S., Chui, C.K.. Coherent Line Drawing. In: Proc. NPAR. ACM; 2007, p. 43–50. doi:10.1145/1274871.1274878.

[33] Winnemöller, H., Kyprianidis, J.E., Olsen, S.C.. XDoG: An eXtended difference-of-Gaussians compendium including advanced image stylization. Computers & Graphics 2012;36(6):740–753. doi:10.1016/j.cag.2012.03.004.

[34] Nienhaus, M., Döllner, J.. Edge-Enhancement – An Algorithm for Real-Time Non-Photorealistic Rendering. Journal of WSCG 2003;11(2):346–353.

[35] Farbman, Z., Fattal, R., Lischinski, D., Szeliski, R.. Edge-Preserving Decompositions for Multi-Scale Tone and Detail Manipulation. ACM Trans Graph 2008;27(3):67:1–67:10. doi:10.1145/1360612.1360666.

[36] Subr, K., Soler, C., Durand, F.. Edge-preserving Multiscale Image Decomposition based on Local Extrema. ACM Trans Graph 2009;28(5):147:1–147:9. doi:10.1145/1661412.1618493.

[37] Kass, M., Solomon, J.. Smoothed Local Histogram Filters. ACM Trans Graph 2010;29(4):100:1–100:10. doi:10.1145/1833351.1778837.

[38] Gastal, E.S.L., Oliveira, M.M.. Domain Transform for Edge-Aware Image and Video Processing. ACM Trans Graph 2011;30(4):69:1–69:12. doi:10.1145/2010324.1964964.

[39] He, K., Sun, J., Tang, X.. Guided Image Filtering. IEEE Trans Pattern Anal Mach Intell 2013;35(6):1397–1409. doi:10.1109/TPAMI.2012.213.

[40] Karacan, L., Erdem, E., Erdem, A.. Structure-preserving Image Smoothing via Region Covariances. ACM Trans Graph 2013;32(6):176:1–176:11. doi:10.1145/2508363.2508403.

[41] Cho, H., Lee, H., Kang, H., Lee, S.. Bilateral Texture Filtering. ACM Trans Graph 2014;33(4):128:1–8. doi:10.1145/2601097.2601188.

[42] Klein, A.W., Li, W., Kazhdan, M.M., Corrêa, W.T., Finkelstein, A., Funkhouser, T.A.. Non-photorealistic Virtual Environments. In: Proc. ACM SIGGRAPH. ACM; 2000, p. 527–534. doi:10.1145/344779.345075.

[43] Praun, E., Hoppe, H., Webb, M., Finkelstein, A.. Real-Time Hatching. In: Proc. ACM SIGGRAPH. ACM; 2001, p. 581–586. doi:10.1145/383259.383328.

[44] Bénard, P., Bousseau, A., Thollot, J.. Dynamic Solid Textures for Real-time Coherent Stylization. In: Proc. ACM I3D. ACM; 2009, p. 121–127. doi:10.1145/1507149.1507169.

[45] Saito, T., Takahashi, T.. Comprehensible Rendering of 3-D Shapes. In: Proc. ACM SIGGRAPH. ACM; 1990, p. 197–206. doi:10.1145/97880.97901.

[46] Döllner, J., Kyprianidis, J.E.. Approaches to Image Abstraction for Photorealistic Depictions of Virtual 3D Models. In: Cartography in Central and Eastern Europe. Springer; 2010, p. 263–277. doi:10.1007/978-3-642-03294-3_17.

[47] Redmond, N., Dingliana, J.. Adaptive Abstraction of 3D Scenes in Real-Time. In: Eurographics Short Papers. The Eurographics Association; 2007, p. 77–80.

[48] Magdics, M., Sauvaget, C., García, R.J., Sbert, M.. Post-processing NPR Effects for Video Games. In: Proc. ACM VRCAI. ACM; 2013, p. 147–156. doi:10.1145/2534329.2534348.

[49] Gibson, J.J.. The Ecological Approach to Visual Perception. Routledge; 1986.

[50] Howard, I.P., Rogers, B.J.. Perceiving in Depth, Volume 3: Other Mechanisms of Depth Perception. 29; Oxford University Press; 2012.

[51] Engel, J., Semmo, A., Trapp, M., Döllner, J.. Evaluating the Perceptual Impact of Rendering Techniques on Thematic Color Mappings in 3D Virtual Environments. In: Proc. Vision, Modeling & Visualization. The Eurographics Association; 2013, p. 25–32. doi:10.2312/PE.VMV.VMV13.025-032.

[52] Lake, A., Marshall, C., Harris, M., Blackstein, M.. Stylized Rendering Techniques for Scalable Real-time 3D Animation. In: Proc. NPAR. ACM; 2000, p. 13–20. doi:10.1145/340916.340918.

[53] Barla, P., Thollot, J., Markosian, L.. X-toon: An Extended Toon Shader. In: Proc. NPAR. ACM; 2006, p. 127–132. doi:10.1145/1124728.1124749.

[54] Furnas, G.W.. Generalized Fisheye Views. In: Proc. CHI. ACM; 1986, p. 16–23. doi:10.1145/22627.22342.

[55] Shneiderman, B.. The Eyes Have it: A Task by Data Type Taxonomy for Information Visualizations. In: Proc. IEEE Symposium on Visual Languages. IEEE; 1996, p. 336–343. doi:10.1109/VL.1996.545307.

[56] Cong, L., Tong, R., Dong, J.. Selective Image Abstraction. Vis Comput 2011;27(3):187–198. doi:10.1007/s00371-010-0522-2.

[57] Kosara, R., Miksch, S., Hauser, H.. Semantic Depth of Field. In: Proc. IEEE InfoVis. IEEE; 2001, p. 97–104. doi:10.1109/INFVIS.2001.963286.

[58] Trapp, M., Beesk, C., Pasewaldt, S., Döllner, J.. Interactive Rendering Techniques for Highlighting in 3D Geovirtual Environments. In: Proc. 3D GeoInfo Conference. Springer; 2010,doi:10.1007/978-3-642-12670-3_12.

[59] Jankowski, J., Hachet, M.. Advances in Interaction with 3D Environments. Comput Graph Forum 2014;in print. doi:10.1111/cgf.12466.

[60] Tominski, C., Gladisch, S., Kister, U., Dachselt, R., Schumann, H.. A Survey on Interactive Lenses in Visualization. In: Proc. EuroVis - STARs. The Eurographics Association; 2014, p. 43–62. doi:10.2312/eurovisstar.20141172.

[61] Lee, B., Isenberg, P., Riche, N., Carpendale, S.. Beyond Mouse and Keyboard: Expanding Design Considerations for Information Visualization Interactions. IEEE Trans Vis Comput Graphics 2012;18(12):2689–2698. doi:10.1109/TVCG.2012.204.

[62] Robles-De-La-Torre, G.. The Importance of the Sense of Touch in Virtual and Real Environments. IEEE MultiMedia 2006;13(3):24–30. doi:10.1109/MMUL.2006.69.

[63] Knoedel, S., Hachet, M.. Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance. In: Proc. IEEE 3DUI. IEEE; 2011, p. 75–78. doi:10.1109/3DUI.2011.5759220.

[64] Isenberg, P., Isenberg, T., Hesselmann, T., Lee, B., von Zadow, U., Tang, A.. Data Visualization on Interactive Surfaces: A Research Agenda. IEEE Comput Graph Appl 2013;33(2):16–24. doi:10.1109/MCG.2013.24.

[65] Keefe, D., Isenberg, T.. Reimagining the Scientific Visualization Interaction Paradigm. IEEE Trans Vis Comput Graphics 2013;46(5):51–57. doi:10.1109/MC.2013.178.

[66] Frisken, S.F., Perry, R.N., Rockwood, A.P., Jones, T.R.. Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics. In: Proc. ACM SIGGRAPH. ACM; 2000, p. 249–254. doi:10.1145/344779.344899.

[67] Bier, E.A., Stone, M.C., Pier, K., Buxton, W., DeRose, T.D.. Toolglass and Magic Lenses: The See-through Interface. In: Proc. ACM SIGGRAPH. ACM; 1993, p. 73–80. doi:10.1145/166117.166126.

[68] Viega, J., Conway, M.J., Williams, G., Pausch, R.. 3D Magic Lenses. In: ACM UIST. ACM; 1996, p. 51–58. doi:10.1145/237091.237098.

[69] Neumann, P., Isenberg, T., Carpendale, S.. NPR Lenses: Interactive Tools for Non-photorealistic Line Drawings. In: Proc. Smart Graphics. Springer; 2007, p. 10–22. doi:10.1007/978-3-540-73214-3_2.

[70] Trapp, M., Glander, T., Buchholz, H., Döllner, J.. 3D Generalization Lenses for Interactive Focus + Context Visualization of Virtual City Models. In: Proc. IEEE IV. IEEE; 2008, p. 356–361. doi:10.1109/IV.2008.18.

[71] Kolbe, T.H.. Representing and Exchanging 3D City Models with CityGML. In: Proc. Int. Workshop on 3D Geo-Information. Springer;

2009, p. 15–31. doi:10.1007/978-3-540-87395-2_2.
[72] Myers, K., Bavoil, L.. Stencil Routed A-Buffer. In: ACM SIGGRAPH Sketches. ACM; 2007, p. 21. doi:10.1145/1278780.1278806.
[73] Nienhaus, M., Döllner, J.. Blueprints: Illustrating Architecture and Technical Parts Using Hardware-accelerated Non-photorealistic Rendering. In: Proc. Graphics Interface. AK Peters; 2004, p. 49–56.
[74] Shanmugam, P., Arikan, O.. Hardware Accelerated Ambient Occlusion Techniques on GPUs. In: Proc. ACM I3D. ACM; 2007, p. 73–80. doi:10.1145/1230100.1230113.
[75] Luft, T., Colditz, C., Deussen, O.. Image Enhancement by Unsharp Masking the Depth Buffer. ACM Trans Graph 2006;25(3):1206–1213. doi:10.1145/1141911.1142016.
[76] Porter, T., Duff, T.. Compositing Digital Images. SIGGRAPH Comput Graph 1984;18(3):253–259. doi:10.1145/964965.808606.
[77] Cockburn, A., Karlson, A., Bederson, B.B.. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. ACM Comput Surv 2009;41(1):2:1–2:31. doi:10.1145/1456650.1456652.
[78] Shneiderman, B., Plaisant, C., Cohen, M., Jacobs, S.. Designing the User Interface: Strategies for Effective Human-Computer Interaction. Pearson; 2009.
[79] Raskin, J.. The Humane Interface: New Directions for Designing Interactive Systems. Addison-Wesley Professional; 2000.
[80] Knoedel, S., Hachet, M., Guitton, P.. Interactive Generation and Modification of Cutaway Illustrations for Polygonal Models. In: Smart Graphics. Springer Berlin Heidelberg; 2009, p. 140–151. doi:10.1007/978-3-642-02115-2_12.
[81] Deriche, R.. Recursively Implementing the Gaussian and its Derivatives. In: Proc. ICIP. IEEE; 1993, p. 263–267.
[82] Nehab, D., Maximo, A., Lima, R.S., Hoppe, H.. GPU-Efficient Recursive Filtering and Summed-Area Tables. ACM Trans Graph 2011;30:176:1–176:12. doi:10.1145/2024156.2024210.
[83] Cao, T.T., Tang, K., Mohamed, A., Tan, T.S.. Parallel Banding Algorithm to Compute Exact Distance Transform with the GPU. In: Proc. ACM I3D. ACM; 2010, p. 83–90. doi:10.1145/1730804.1730818.
[84] Smith, K., Landes, P.E., Thollot, J., Myszkowski, K.. Apparent Greyscale: A Simple and Fast Conversion to Perceptually Accurate Images and Video. Comput Graph Forum 2008;27(2):193–200. doi:10.1111/j.1467-8659.2008.01116.x.
[85] DeCoro, C., Cole, F., Finkelstein, A., Rusinkiewicz, S.. Stylized Shadows. In: Proc. NPAR. ACM; 2007, p. 77–83. doi:10.1145/1274871.1274884.
[86] Rosenholtz, R., Li, Y., Nakano, L.. Measuring Visual Clutter. Journal of Vision 2007;7(2):1–22. doi:10.1167/7.2.17.
[87] Harel, J., Koch, C., Perona, P.. Graph-Based Visual Saliency. Advances in Neural Information Processing Systems 2007;19:545–552.