Journal of Location Based Services
Vol. 00, No. 00, Month 200x, 1–16

Image-Based Strategies for Interactive Visualization of Complex 3D Geovirtual Environments on Lightweight Devices

Dieter Hildebrandt*, Benjamin Hagedorn, and Jürgen Döllner

Hasso-Plattner-Institut, University of Potsdam, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

(*v1.0 Submitted 30th October 2010*)

In this paper, we present strategies for service-oriented, standards and image-based 3D geovisualization that have the potential to provide interactive visualization of complex 3D geovirtual environments (3DGeoVE) on lightweight devices. In our approach, interactive geovisualization clients retrieve sets of 2D images of projective views of 3DGeoVEs generated by a 3D rendering service. As the key advantage of the image-based approach, the complexity that a client is exposed to for displaying a visual representation is reduced to a constant factor primarily depending on the image resolution. To provide users with a high degree of interactivity, we propose strategies that are based on additional service-side functionality and on exploiting multiple layers of information encoded into the images for the local reconstruction of visual representations of the remote 3DGeoVE. The use of service-orientation and standards facilitates designing distributed 3D geovisualization systems that are open, interoperable and can easily be adapted to changing requirements. We demonstrate the validity of the proposed strategies by presenting proof-of-concept implementations of several image-based 3D clients for the case of virtual 3D city models.

Keywords: 3D geovirtual environments; Distributed 3D geovisualization; Image-based representations; Lightweight devices; Service-oriented architectures; Standardization

1 Introduction

For the interactive 3D geovisualization of complex 3D geovirtual environments (3DGeoVE) such as virtual 3D city models and landscape models, massive amounts of geodata as well as complex processing and visualization algorithms are involved. For interactive access to these models, the amount of required resources for generating visual representations in terms of network, storage, and computing capacity significantly reduce the applicability of 3D geovisualization especially on mobile devices. As a common solution, visualization systems can be deployed that distribute geodata and functionality over computers connected by a network using visualization clients (e.g., Google Earth). However, common approaches for distributed visualization either do not scale with the increasing complexity of geodata (e.g., streaming detailed, textured CAD-based 3D city models) or the computation required for visualization (e.g., real-time photorealistic 3D rendering), do not easily scale with an increasing number of concurrent users, provide only limited interactivity, or yield closed, tightly coupled systems.

In this paper, we present strategies for service-oriented, standards and image-based 3D geovisualization that have the potential to overcome the aforementioned limitations. In our approach, interactive geovisualization clients retrieve a set of 2D images of projective views of 3DGeoVE generated by a 3D rendering service. As the key advantage of the image-based approach, the complexity that a client is exposed to for displaying a visual representation is reduced to a constant factor primarily depending on the image resolution. To provide users with a high degree of interactivity, we propose strategies that are based on additional service-side functionality and on exploiting multiple layers of information encoded into the images for the local reconstruction of the remote 3DGeoVE. The use of service-orientation and standards facilitates designing distributed 3D geovisualization systems that are open, interoperable and can easily be

*Corresponding author. Email: dieter.hildebrandt@hpi.uni-potsdam.de

Journal of Location Based Services
ISSN 1748-9725 print / ISSN 1748-9733 online
http://www.tandf.co.uk/journals
DOI: 10.1080/1748972YYxxxxxxxx

adapted to changing requirements. We demonstrate the validity of the proposed strategies by presenting proof-of-concept implementations of several image-based 3D clients for the case of virtual 3D city models.

The remainder of this paper is structured as follows. In Section 2, we identify a set of requirements for a specific class of practically relevant 3D geovisualization systems. The fundamentals of our approach including SOA, standards, and the distributed visualization pipeline as well as related work are described in Section 3. We present the outline of the general approach we propose for designing 3D geovisualization systems intended to meet the previously identified requirements in Section 4. As instances of the general approach, we present three concepts for image-based, interactive visualization clients in Section 5. In Section 6, we discuss how the proposed approach and the three concrete concepts support meeting the previously identified requirements. Finally, in Section 7 we conclude this paper with a summary, conclusions, and future work.

2 Requirements

In this Section, we identify a set of requirements for 3D geovisualization systems. This particular set is valid for a specific, practically relevant class of 3D geovisualization systems and is informed by existing literature. In this paper, we place a particular focus on 3DGeoVEs and virtual 3D city models. Furthermore, we focus on 3D rendering for generating 2D images of projective views of primarily static CAD-based models with real-time interaction and navigation using six degrees of freedom.

(i) Support for *integration* (R1) is required to connect computer systems effectively and efficiently on different levels of abstraction such as data, functionality, process, visualization, interaction, and system. It should improve the flexibility and efficiency of adapting systems to changing requirements and ease the reuse of software components (Rhyne and MacEachren, 2004; Brodlie et al., 2007).

(ii) *Interoperability* (R2) increases the effectiveness and efficiency of the integration on the different levels and can be improved by applying standards. In the geospatial and the geovisualization domain, insufficient interoperability has been identified as a major barrier for progress in the respective domain (Bishr, 1998; MacEachren and Kraak, 2001; Andrienko et al., 2005).

(iii) Typically, in real world applications, systems are required to facilitate processing, visualizing and interacting with *massive amounts of geodata* (R3). In particular, this applies to virtual 3D city models (MacEachren and Kraak, 2001; Hildebrandt and Döllner, 2009).

(iv) Providing *effective, high quality visual representations* (R4) improves the effectiveness of a geovisualization system and is facilitated by advanced, complex, innovative visualization algorithms and in certain cases massive amounts of data (e.g., for virtual 3d city models), for both realistic and abstract views (Döllner, 2005; Hildebrandt and Döllner, 2009).

(v) Support for *platform independency* (R5) comprises the relative independency of a system solution from software and hardware platforms on different levels of abstraction and adaptive and moderate use of platform resources. It can improve dissemination and reduce costs (MacEachren et al., 2004; Brodlie et al., 2007).

(vi) A *high degree of interactivity* (R6) is a key defining characteristic and as well as a crucial requirement for geovisualization systems and should be effective and efficient (MacEachren and Kraak, 2001; Dykes, 2005).

(vii) Support for *styling* (R7) visual representations allows control over what (e.g., filtering of features) and how to portray (e.g., mapping of features to geometries and visual attributes) and is essential for interaction and generating different visualizations from the same base data (Yi et al., 2007; Neubauer and Zipf, 2009).

In the following Sections, we make explicit reference to each introduced requirement via its respective code (e.g., "R1" for the first listed requirement) when the discussion touches the requirement.

3 Fundamentals and Related Work

3.1 *SOA, Standards, and the Distributed Visualization Pipeline*

The *service-oriented computing* (SOC) paradigm promotes the idea of assembling application components into a network of services that can be loosely coupled to create flexible, dynamic business processes and agile applications that span organizations and computing platforms (Papazoglou et al., 2007). The term *service-oriented architecture* (SOA) denotes both an architectural concept and style that adheres to the SOC paradigm and concrete architectures that are designed following that architectural concept. SOC and SOA are specific paradigms for designing *distributed systems*.

In the geospatial domain, the Open Geospatial Consortium (OGC, (OGC, 2010)) adopted the SOA paradigm and proposes standards for service interfaces, data models, and encodings. For the presentation of information to humans, the OGC proposes stateless *portrayal services*. For 3D portrayal, the *Web 3D Service* (W3DS) (Schilling and Kolbe, 2010) and the *Web View Service* (WVS) (Hagedorn et al., 2009, 2010) are proposed as different approaches that are both still in the early stages of the standardization process. The major difference in the current proposals for the 3D portrayal services is the representation that they generate and what visualization pipeline stages they implement to what extent. The W3DS delivers scene graphs that can be rendered by a client, whereas the WVS delivers rendered images of projected views that are ready for display. An analysis of the respective strengths and weaknesses of 3D portrayal services can be found in (Hildebrandt and Döllner, 2009). As a complement to this, the *Styled Layer Descriptor* (SLD) and *Symbology Encoding* (SE) (Lupp, 2007; Neubauer and Zipf, 2009) are standardization proposals for user-defined styling of 2D and 3D visual representations. Portrayal services may include support for SLD.

We introduce the visualization pipeline as a model for allocating resources in a distributed system and for motivating the image-based representation. The *visualization pipeline* (Haber and McNabb, 1990) is a well-established concept for separating the concerns of the process of generating visual representations from data in three stages. The *data* is *filtered* into enhanced data, then *mapped* to visualization objects (e.g., represented as scene graphs, geometry, and visual attributes), and finally *rendered* into a digital 2D image that is ready for *display* to a human user. For designing a geovisualization system based on SOA, the visualization pipeline must be functionally decomposed. A basic, conceptual decomposition splits the pipeline into two parts interconnected by a network, resulting in a 2-tier physical client/service architecture. The separation can be applied after the filtering, mapping, or rendering stage. Three types of geovisualization clients can be categorized: *thick clients*, *medium clients*, and *thin clients* (adapted from (Doyle and Cuthbert, 1998)). Note that this classification is schematic. Concrete systems may implement variations of this model, as demonstrated in Sections 5.2 and 5.3. The W3DS and WVS adhere to this model. The W3DS provides scene graphs as output of the mapping stage, whereas the WVS provides images as output of the rendering stage.

3.2 *Related Work*

In this paper, we are concerned with strategies for service-oriented, standards and image-based 3D geovisualization systems that support meeting the requirements identified in Section 2.

For distributed visualization systems, most commonly visual representations based on scene graphs, geometry such as triangle meshes, and texture maps are proposed and applied (such as in the W3DS (Schilling and Kolbe, 2010), Google Earth, Microsoft Bing Maps 3D). Here, we focus on image-based representations since we estimate that they have the potential to better support the stated requirements. For image-based, distributed visualization, the proposed approaches include streaming videos of rendered 3D models from a server to a tightly coupled client (Lamberti and Sanna, 2007), applying *image-based modeling and rendering* (IBMR) (Shum et al., 2007) and warping a representation based on color and depth images retrieved from a remote server for rendering novel views (Chang and Ger, 2002), applying *point-based modeling and rendering* (PBMR) (Gross and Pfister, 2007) and utilizing remotely rendered color and depth images as input for client-side PBMR (Ge, 2007), and rendering novel views on the client by warping between image-based panoramas retrieved from a server (Filip, 2009). In addition, proposals exist for designing

visualization systems as distributed systems (e.g., (Brodlie et al., 2004)), distributed systems based on SOA (e.g., (Wang et al., 2008)), or distributed systems based on SOA and OGC standards (e.g., (OGC, 2010; Hildebrandt and Döllner, 2009; Basanow et al., 2008)).

However, to the best of our knowledge, we are not aware of related work that proposes designing 3D geovisualization systems based on SOA, OGC standards, and image-based representations with the aim of meeting the previously stated requirements. In particular, related work often does not address at the same time improving integration by loose coupling, interoperability, supporting lightweight clients, and the application to 3DGeoVE.

4 General Approach

We propose a particular approach for designing 3D geovisualization systems that is intended to support meeting the requirements identified in Section 2. In this Section, we outline the general approach. Based on the general approach, we present three different, concrete concepts in the following Section 5. The three presented concepts differ in the degree that they meet the stated requirements.

4.1 *Working Principle*

The general approach is based on the distributed visualization pipeline, image-based representations, standards, and SOA (Fig. 1). The 3D rendering service implements all stages of the visualization pipeline and locally stores the geodata that it can portray. Clients retrieve 2D images of projective views of a 3DGeoVE from the service. Clients then either directly display the retrieved images, or use the images as input for further processing. Interactions of users with the graphical user interface (GUI) of the display result in user input events. The *controller* (CTRL) and its implemented interaction techniques process these events. The *view process* (VP) transforms commands for updating the visualization from the controller into the calling of service operations for requesting images or other functionality. In Section 5, we present three client concepts that differ in how they exploit images retrieved from the service and use additional service-side functionality for providing interactivity. We apply standards where available and feasible. The 3D rendering service implements the WVS service interface, images are encoded using standard image formats (i.e., JPEG, PNG), clients access the service via HTTP on top of TCP/IP, and if a client requests multiple images from the service with one call, the service returns images as one multipart response. The architecture and service complies with design principles commonly proposed for SOA (Erl, 2005). The WVS is, e.g., stateless, loosely coupled, and autonomous.

As service interface for the 3D rendering service, we propose the OGC standardization proposal WVS (Hagedorn et al., 2009, 2010). The WVS overcomes restricted visualization and interaction capabilities of preceding proposals such as the *Web Terrain Service* (WTS, OGC discussion paper) and its successor the *Web Perspective View Service* (WPVS, OGC-internal draft specification). The OGC approved the WVS specification as discussion paper. It provides a) additional image layers for 3D views, and b) additional service operations for supporting analysis, navigation, and information retrieval. The WVS supports retrieving additional image layers for 3D views besides color layers through the `GetView` operation. These layers store various spatial and thematic information for each image pixel such as color, spatial depth, object ID, surface normal, and mask. This concept is based on the *G-buffer* (Saito and Takahashi, 1990) concept from 3D computer graphics. This data does not necessarily represent color values and is not necessarily dedicated for human cognition. However, the WVS supports encoding also non-color image layers by standard image formats. The result is that for all image layers the same principles for data encoding, data exchange, and client-side data loading and processing can be applied. Additionally, using image encodings allows for applying state-of-the-art compression algorithms. One implication is that on the service-side for each pixel each non-color information has to be encoded as a color (e.g., with four components as RGBA). Symmetrically, the client has to decode non-color information from color. As most important additional operations, the WVS provides the following: `GetFeatureInfo` (returning attribute information for a feature identified by a specified 2D coordinate in a 3D view), `GetPosition` (returning

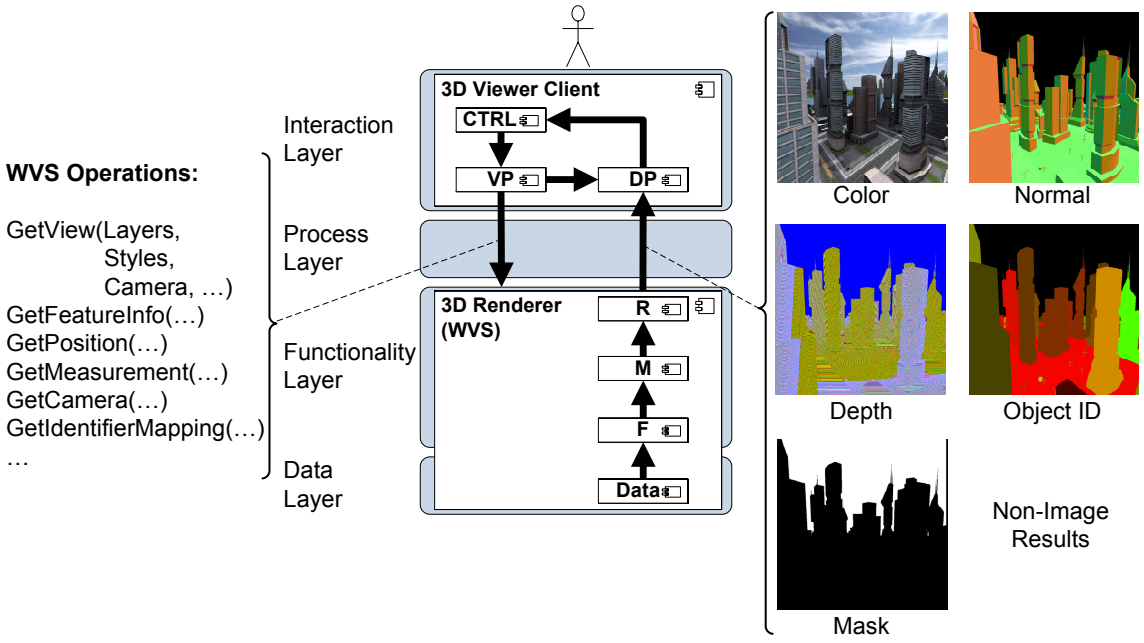


Figure 1.: Architecture of the general approach. Clients use the operations provided by the WVS to retrieve image layers and other results.

the 3D position of a part of a feature specified by 2D coordinate in a 3D view or the 2D coordinate of a specified 3D position), `GetMeasurement` (returning the Euclidean length of a path or the area of a polygon specified by a set of 2D coordinates in a 3D view), `GetCamera` (returning a camera specification providing a "good view" on features identified by 2D coordinates in a 3D view), and `GetIdentifierMapping` (returning mappings between GML feature IDs and object IDs that encode the GML feature IDs compactly as integer pixels in object ID layers). Based on these functional extensions, WVS clients can implement various 3D visualization and interaction features without changing the underlying working principle. This leads to an increased degree of interactivity and is demonstrated by prototypic web-based client applications in Section 5.

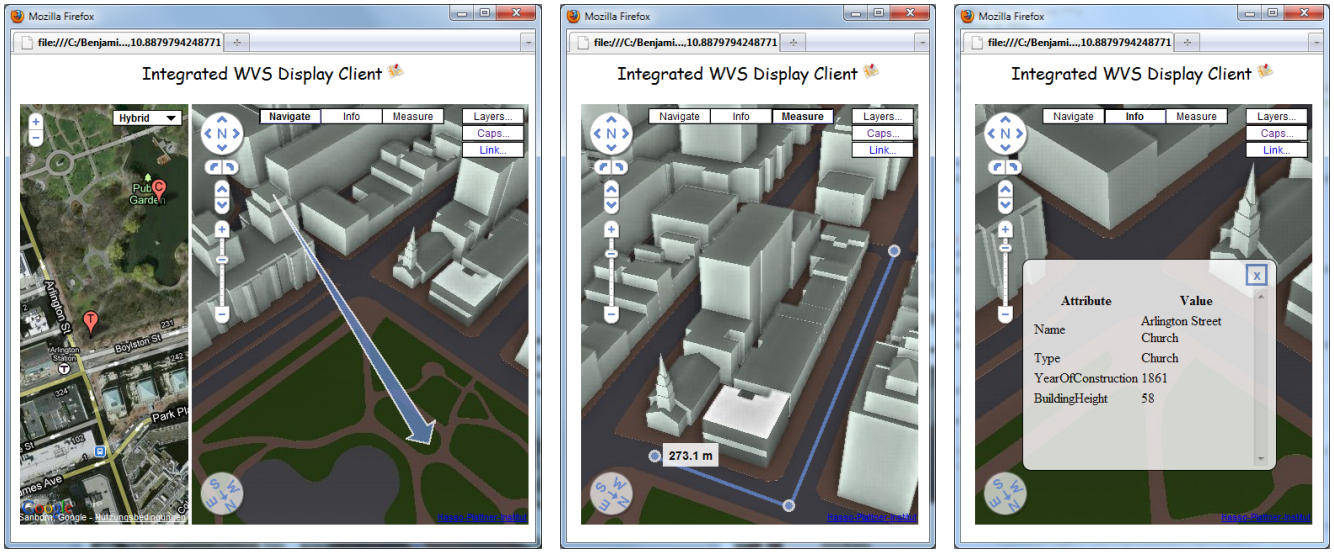
4.2 *Challenges*

However, there are fundamental challenges when applying image-based representations regarding interactivity (R6) and the efficient use of the network channel in the course of interactions (R3).

Interaction occurs by manipulating parameters of the visualization pipeline stages that results in updated displays. Separating the rendering and display of images by a network introduces the high latency and low bandwidth of the network channel to the interaction loop. If real-time navigation is required (i.e., requiring display updates with more than 10 frames per second, and six degrees of freedom), this results in displays with low or limited frame rates and high latencies between user input and display updates. This drawback also applies to interaction techniques that change parameters of the rendering or preceding stages (e.g., removing or changing the styling of features).

Furthermore, interaction techniques that require access to features of a model and their properties can be limited by a purely image-based approach (e.g., highlighting, relating, or showing additional information for features). For implementing these techniques supplying merely color does not provide sufficient information. Access to the outputs of previous stages is required.

One single view of a 3DGeoVE based on massive geodata is typically most efficiently encoded as an image. However, in the course of interaction numerous images are required (e.g., when navigating a virtual camera). Basic image-based approaches instantly discard an image after it has been displayed and replace it with a newly rendered, self-contained image. Not existing or limited reuse of image data can significantly reduce memory efficiency and increase interaction latency. In addition, service load increases.



(a) (b) (c)

Figure 2.: Screenshot of a web-based WVS display client. (a) Integrated with a 2D map from Google maps, which marks camera look-to and look-from. The arrow indicates the selection of a new camera look-from and look-to within the image. (b) Specification of a path and display of its length. (c) Display of thematic information retrieved from the WVS for a feature selected by the user. (3D data: Boston Redev. Authority)

5 Concepts for Image-Based, Interactive Visualization Clients

In this Section, we present three concepts for image-based, interactive visualization clients. The first is based on additional service-side functionality, while the others are based on local 3D reconstruction of the remote 3DGeoVE. The presented concepts are intended to meet the requirements identified in Section 2. Each concept is an instance of the general approach presented in Section 4, and, in particular, is intended to tackle the challenges of the general approach identified in Section 4.2.

For each concept, we briefly present a prototypical, rudimentary proof-of-concept implementation. Moreover, we evaluate how each concept provides interactivity by examining how it supports specific interaction categories. We employ the following seven categories that are proposed in (Yi et al., 2007) and are based on the notion of user intent: *Select* (mark something as interesting), *Explore* (show me something else), *Reconfigure* (show me a different arrangement), *Encode* (show me a different representation), *Abstract/Elaborate* (show me more or less detail), *Filter* (show me something conditionally), and *Connect* (show me related items).

5.1 *Concept Based on Image Retrieval and Display*

The WVS display client presented in this Subsection conceptually requests 3D views from a WVS as color images and directly displays these images (Fig. 2). As a complement to this, it allows users to control the virtual camera, to retrieve information about displayed features, and to perform analysis in the displayed 3DGeoVE. For this, the client takes advantage of various WVS operations that are designed for supporting interactivity even on thin clients.

Technically, the client is a JavaScript-based web application, which is fully executed on the client side. Thus, technical barriers for its application are low. It runs in any web browser that supports JavaScript. No additional plug-ins (e.g., Java or Flash) need to be installed and no dedicated 3D rendering hardware or software is required at the client side. Due to this, the WVS display client is particularly applicable on platforms that are limited in computing and 3D rendering capabilities or are faced with limited network connectivity (e.g., mobile phones). Furthermore, the client can be easily integrated into existing web sites and web applications.

For maneuvering the camera, the WVS display client determines a new camera specification and requests a new view from the WVS. Currently, no additional intermediate views are considered. Thus, camera control is conceptually not continuous, but inherently discrete and step-by-step. The camera can be manipulated by a) GUI controls (translate, rotate left/right, tilt up/down, zoom in/out, orient to north), b) mouse-wheel usage modified by keys (zooming, rotating), and c) selection of one or multiple 2D positions in the displayed view.

The client supports *in-image interaction tools* as a major concept to allow users to interact directly with the 3D view and the contained features. For this, several WVS operations are available that require one or more image pixels as input. For in-image camera control, the client transforms one or more selected pixel positions into a corresponding 3D geospatial location by the WVS `GetPosition` operation (implemented service-side as ray intersection tests). The returned 3D locations can be used within new `GetView` requests as new camera positions and/or orientations. Further, the client incorporates the `GetCamera` operation to utilize service-side support for smart camera control: For a specific 2D pixel input, a WVS can compute "good" camera specifications. This enables assisting and higher-level 3D camera control for thin clients, including input preprocessing (e.g., sketch recognition) as well as the consideration of the type and geometry of affected features.

To foster information gathering, the client can retrieve and display information for selected features (implemented through the `GetFeatureInfo` operation). Additionally, the client allows users to perform distance and path measurements within the displayed view. This is based on the `GetMeasurement` operation, a generic approach for providing analysis functionalities that is also based on 2D pixel positions. Measurements are computed at the service-side and returned to the client for display.

Using HTML and a JavaScript drawing library, the client can annotate the 3D view by text or drawings, for integrating, e.g., feature information or visual navigation feedback. Examples are overlays marking interesting or selected positions, arrows indicating a new camera look-from and look-to, and paths that are measured.

In summary, the WVS display client supports the interaction categories presented in Section 5 as follows. For *Select*, clicking on a feature generates a mark at this position. For *Explore*, the client allows for rotating at the camera position, tilting the camera up/down, translating the camera, orienting the camera towards a selected location of interest, as well as moving the camera to a location picked in the image. Points of interest can be selected from a drop-down-menu. For *Reconfigure*, the client allows for rotating around the camera's look-to, and for moving the camera up and down while keeping the look-to. For *Abstract/Elaborate*, the client allows for zooming into the scene as well as for showing feature information in an information widget. To *Filter* a user can select the data layers to select from a GUI control. For *Encode*, a user can select the visual style to apply to the selected data from a GUI control. Finally, to *Connect*, the display client can be integrated and combined with other views showing the related data (Fig. 2).

5.2 *Concept Based on Image-based Modeling and Rendering*

In this Subsection, we present a concept that employs a latency hiding technique based on a client-side, partial visualization pipeline and 3D reconstruction of the visual representation of the remote 3DGeoVE from images. For the client-side 3D reconstruction and rendering, the concept employs techniques based on image-based modeling and rendering (IBMR).

5.2.1 ***Latency Hiding Technique.*** As a general strategy to mitigate the negative effects on interactivity of high latency and low bandwidth introduced by a distributed visualization pipeline, we first propose *avoiding* the execution of visualization pipeline stages. Instead of aiming at reducing the absolute latency of the system, we aim at reducing the user-perceived latency (Sisneros et al., 2007). This can be achieved by using latency hiding techniques that trade steadily low response times with approximations.

We propose a concrete latency hiding technique that is based on the nested, partial visualization pipeline architectural pattern and the client-side, partial 3D reconstruction of the visual representation of the

remote 3DGeoVE from images. A pattern that extends the visualization pipeline and that we observe in existing systems is what we term the *nested, partial visualization pipeline* (NPVP). Using this pattern, partial pipelines can be nested in the main pipeline after the filter, map, or render stage (e.g., F(FMR) M(FMR) R(FMR)). Stages must possibly reinterpret the output of a preceding stage (e.g., rendered images as textures for subsequent rendering in FMR(R)). We use the NPVP pattern to insert a partial pipeline in the main pipeline after the rendering stage that consists of an additional mapping and rendering stage (Fig. 3). As before, the WVS implements a filtering, mapping, and rendering stage and the client retrieves images from the WVS.

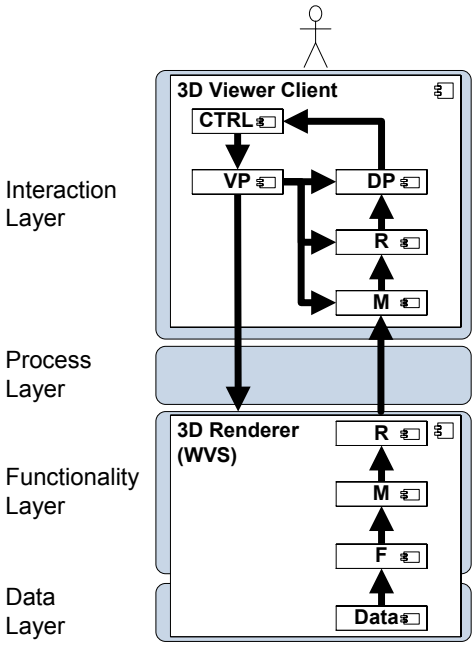


Figure 3.: Architecture employed for the proposed latency hiding technique used by both the concept based IBMR and the one based on PBMR.

In the newly inserted mapping stage on the client, the images are reinterpreted as a 3D visual representation of the remote 3DGeoVE. Each pixel is conceptually interpreted as a surface patch in 3D space that covers a part of the visible surface of the 3DGeoVE with attributes including 3D extend, color, and object ID. From the surface patches, a computer graphics representation based on geometry and visual attributes is constructed. The mapping stage provides a mechanism for aggregating multiple, consecutively retrieved image sets that depends on the type of representation used. The aggregated representations constitute the client-side, partial 3D reconstruction of the visual representation of the remote 3DGeoVE.

In the newly inserted rendering stage, novel views can be rendered of the local 3D reconstruction from arbitrary virtual camera viewpoints. For a specific view, the available 3D reconstruction on the client is typically under- or oversampled in comparison to the available original data in the distributed pipeline. For this reason, the images rendered from the reconstruction represent only approximations of the visual representations of the remote 3DGeoVE. By using the interaction techniques the client provides, a user can manipulate parameters of the pipeline stages. Manipulating the local pipeline results in low latency updates of the display (e.g., for real-time navigation with six degrees of freedom). Specific interaction techniques require manipulating parameters of the remote pipeline (e.g., for changing the styling, or when parts of the model come into view that were not sufficiently sampled with previously retrieved images). These manipulations still result in high latency responses. Effectively, decoupling the update of the display from the high latency, distributed pipeline, and introducing a local, low latency pipeline based on the NPVP pattern and 3D reconstruction of the remote 3DGeoVE allows hiding the absolute latency and updating the display with low latency.

5.2.2 ***Image-based Modeling and Rendering.*** In this Subsection, we present an instance of the general strategy that is based on techniques from the domain of *image-based modeling and rendering* (Shum et al., 2007). Figure 4 (left) depicts a screenshot of an implementation of this concept.

In the mapping stage, the client retrieves for a given camera specification a set of perspective images consisting of a color, depth, and object ID image layer. From a depth image and the camera specification it was retrieved with, a 3D depth triangle mesh is constructed. First, each depth value is projected back to a 3D coordinate in camera space and then transformed into world space. Second, for every original pixel in the depth image two triangles are created that connect its corresponding 3D coordinate with the coordinates of three neighboring pixels (i.e., for pixel coordinate (x,y) two triangles with the following coordinates are created: (x,y), (x+1,y), (x,y+1) and (x+1,y), (x+1,y+1), (x,y+1)). The mapping stage caches each image set and its corresponding mesh until the controller triggers its eviction explicitly or by being the least recently used image set when the memory limit for the cache is exceeded.

In the rendering stage, for a given camera specification a novel view is rendered from the available depth meshes of the mapping stage. The camera specification as an input for this stage is typically provided by an interaction technique applied by a human user. Each depth mesh that spatially intersects the 3D view frustum of the camera is rendered. The rendering applies color to the depth mesh via projectively texturing of the depth mesh with its corresponding color image. It resolves visibility in the frame buffer with via depth buffering. Multiple depth meshes are aggregated in screen space to represent the remote 3DGeoVE from a set of locally available sampled images of the 3DGeoVE. For highlighting a feature identified by an object ID, the depth mesh is rendered by a shader that colors each pixel with the same object ID. Additionally, the contours of features are detected by detecting edges in the object ID image. Feature contours and interiors in image space are colored differently.

The controller receives user input events and implements interaction techniques. The controller is responsible for providing camera specifications as input parameters for the local rendering stage. Furthermore, the controller is responsible for implementing a *sampling strategy*. The sampling strategy controls how the remote 3DGeoVE is sampled by retrieving images and when already present samples can be discarded. The sampling strategy depends on the applied interaction technique and current and assumed future camera specifications. Its goal is to provide for each novel view that is rendered on the client a set of images that allows rendering the view with minimal under- and minimal oversampling. Undersampling displays in the novel view as holes (i.e., no samples are available for an area) or a blurred area (i.e., samples not dense enough). A negative consequence of oversampling (i.e., samples too dense for an area) is overdraw and reduced rendering performance and frame rate.

The implementation of the concept supports the seven interaction categories as follows. As an interaction technique in the category *Select*, clicking on a feature highlights the feature (implemented by coloring all pixels with the same object ID as the clicked on pixel). For *Explore*, the client allows rotating the camera around itself, moving the camera by panning, and moving it by selecting a feature of interest that is then brought into focus via a continuous camera animation (implemented by rendering the local 3D reconstruction of the 3DGeoVE in a local NPVP with low latency). For *Reconfigure*, the client allows rotating the camera on a sphere around a selected feature in the 3DGeoVE (implementation similar to previous category). For *Encode*, the client can specify the styling in a SLD document and retrieve projected images with the styling applied from a WVS. For *Abstract/Elaborate*, the client offers (geometric) zooming and tool-tips for displaying additional information about features (implemented by modifying the field-of-view of the virtual camera and by retrieving additional information encoded in GML for a feature identified by its object ID). To *Filter* the data set being presented, the client can specify in a SLD document conditionally what features to include and retrieve images from WVS with the filtering applied. For *Connect*, the client can be combined with different displays in a system based on coordinated, multiple views (e.g., as part of a mashup as presented in Section 5.1).

5.3 *Concept Based on Point-based Modeling and Rendering*

In this Subsection, we present a second concept that employs the latency hiding technique presented previously in Subsection 5.2.1. However, instead of IBMR, the concept presented in this Section employs techniques based on *point-based modeling and rendering* (PBMR) (Gross and Pfister, 2007) for the client-side 3D reconstruction and rendering. Since both concepts share several similarities, here, we will focus on presenting the differences between them. Figure 4 (middle, right) depicts screenshots of an implementation of this concept executing on a smartphone.

In the mapping stage, the client interprets the 3D surface patches derived from the images as 3D points with the attributes color and object ID. In addition, each point is assigned a spatial extent in object space derived from the surface patch. We assign a radius to each point that effectively interprets a patch as a sphere. Note that other representations such as circular disks, elliptical disks, or voxels could be used. Conceptually, the spheres derived from one image set approximate the continuous surface of the visual representation of the 3DGeoVE visible in the image set. The spheres define the visible surface geometry and topology. Then, the spheres are added to an octree, a hierarchical spatial data structure. Spheres are stored in the nodes and leafs of the octree. In the previously introduced IBMR representation, each image

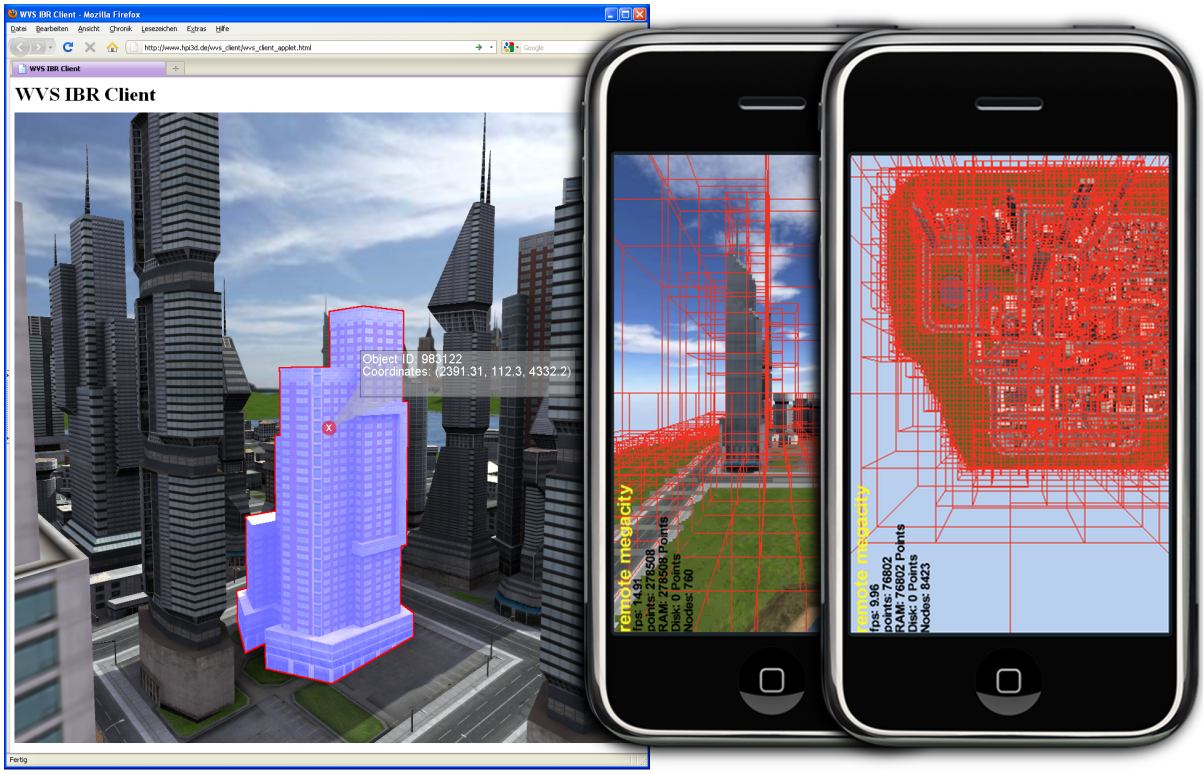


Figure 4.: Left: Screenshot of a web-based WVS IBMR client application (Section 5.2) executing inside a web browser. The IBMR client shows a highlighted and annotated feature that was selected by the user. The IBMR client represents the remote 3D GeoVE locally as sets of 3D depth triangle meshes. Middle, right: Screenshots of a WVS PBMR client application (Section 5.3) executing on the smartphone Apple iPhone. The screenshots depict a close up view of a 3DGeoVE (middle) and an overview (right). The PBMR client represents the remote 3D GeoVE locally as an octree that contains 3D surface patches in nodes and leafs. In the screenshots, the client draws the bounding box of each octree node traversed in the rendering algorithm to illustrate the underlying hierarchical spatial data structure.

set is transformed into one corresponding depth mesh and, thus, the 3D surface patches derived from an image stay in the context of that image. In contrast, in the PBMR representation, the surface patches are disconnected from each other and their originating image set. They are added separately to the octree. The octree is used for aggregating data, storing data in multiple resolutions, querying data, and rendering data. Data received from multiple calls to the WVS is integrated in a unified manner. Multiple resolutions are stored at different levels of the data structure, whereby each node stores a generalized representation of its child nodes. The data structure supports querying spatial data in logarithmic complexity.

We assume that in practical cases the amount of data needed for visualizing 3DGeoVEs exceeds the main memory capacities of targeted computers used for executing the client. Moreover, in contrast to the IBMR representation, the aggregation of the local representation has the potential to be significantly more effective and efficient. To exploit this potential, we aggregate on the client as much data as feasible. For this, we propose to utilize appropriate *out-of-core techniques* (Gobbetti et al., 2008) for implementing the client application. The general strategy is to first retrieve the data as images from the WVS and dynamically add the data to the octree. Then, when the amount of the locally accumulated data exceeds the main memory capacity, parts of the octree are stored on the local hard disk and are removed from main memory. Subsequently, when data is needed that does not reside in main memory but on the hard disk, it is retrieved from that location instead of from the remote service.

In the rendering stage, the client uses the octree for rendering novel views. The octree supports rendering the multi-resolution representation with view frustum culling, level-of-detail control, and control of a trade-off between performance and quality. The rendering algorithm traverses the octree. When the projected screen space area of a node falls below a given threshold (e.g., one pixel, or more than one pixel for coarser

representation and faster rendering), the spheres contained in the node are rendered. When a leaf node is reached, its projected screen space area is generally larger than the threshold and, thus, larger than one pixel. In this case, the contained spheres are rendered with the *splatting* (Gross and Pfister, 2007) technique.

The controller in the PBMR concept has the same role as in the IBMR concept. However, the different behavior of the mapping and rendering stages (e.g., regarding aggregation) require an accordingly adapted sampling strategy. The implementation of the PBMR concept supports the seven interaction categories in a similar way as the implementation of the IBMR concept presented in Section 5.2.

6 Discussion

In this Section, we discuss how the approach proposed in Sections 4 and 5 supports meeting the requirements identified in Section 2.

6.1 *Applying SOA and Standards*

We propose designing 3D geovisualization systems as distributed systems based on SOA and OGC standards. By designing the system as a distributed system, the resources for generating visual representations in terms of network, storage, and computing capacity can be allocated to computers within a network. Thus, local clients and devices are freed from the burden to provide all required resources locally. This can result in lightweight clients that can operate, e.g., in web browsers and on mobile devices (supporting R5). Applying SOA for designing distributed systems has the potential to improve several identified requirements. SOA promotes interface orientation, encapsulation, hiding of implementation details, a common base technology (e.g., web services), and a unified architectural view on the system landscape on a high level of abstraction. These characteristics potentially improve support for integration (R1), interoperability (R2), and platform independency (R5). Moreover, reusable, competing services can encapsulate complex computer graphics and geovisualization concepts, techniques and metaphors to support effective visual representations (R4) (Döllner, 2005). Applying standards on the application level (i.e., OGC standards) and on the base technology level (e.g., from W3C, OASIS, ISO) as available and feasible potentially improves interoperability (R2).

6.2 *Applying Image-based Representations*

In our approach, interactive geovisualization clients retrieve sets of 2D images of projective views of 3DGeoVE generated by the 3D rendering service WVS. We decided to separate the client and the service after the rendering stage and to use standards image formats as interchange formats. We argue that this approach has specific advantages in meeting the stated requirements compared to other common approaches. In particular, this includes approaches that separate client/service after the mapping stage and transfer representations based on scene graphs, geometries, and textures (such as the W3DS).

Applying images for communication offers advantages regarding integration and interoperability (R1, R2). Images are conceptually simple, robust, commonly used and supported. Additionally, image formats exist (e.g., JPEG, PNG) that are standardized, commonly used and supported, and storage and processing efficient. Using the G-buffer concept (Saito and Takahashi, 1990), multiple information layers of a 3D model (e.g., 3D position, normal, color, and object ID of surface elements) can be encoded into 2D images. Thus, images can be used as an alternative representation for 3D models that sample 3D models in a discrete and multidimensional way and reduce the diversity and heterogeneity of their original representations (e.g., points, triangles, NURBS, voxel) to a simpler, unified representation.

Regarding the communication between client and service — which can be considered one major bottleneck — the image-based representation reduces the client/service communication complexity to a constant factor primarily depending on the image resolution (R3). For one requested view, standardized, mature image formats encode explicitly per pixel what is required to reproduce the view with given quality

preferences using specific, highly optimized compression algorithms. The required storage size of an uncompressed image does not depend on the complexity of the original 3D scene. It defines the upper bound for a compressed image representation. In contrast, for instance, the size of representations based on scene graphs directly depends on the complexity of the 3D scene. For visualizing massive amounts of geodata, typically, the storage required for encoding one view as an image is significantly smaller than encoding it as a scene graph. Transferring representations based on scene graphs to clients puts practical limits on the complexity of the models that can be accessed and portrayed. These limits are significantly lower when applying the image-based representation.

Furthermore, using image-based representations has the potential to better support effective, high quality visualization (R4) and platform independency (R5). Visual representations presented to users do not significantly depend on client resources such as storage and computing capacities. In particular, visualizations are not restricted by the requirement that the employed 3D rendering is compatible with every hardware and software configuration that potential users could provide. Instead, geodata and processing reside in controlled, potentially powerful server environments. This allows lightweight clients that can operate, e.g., in web browsers and on mobile devices. Furthermore, visualizations are not limited by the expressiveness of an intermediate, standardized description of the visual representation. For instance, when using a scene graph representation, clients are expected to render the model as specified in the scene graph. Since common scene graph representations (e.g., VRML, X3D, KML) cannot express every visualization and rendering technique that is applied in the 3D geovisualization domain, their expressiveness is limited. When using a 3D rendering service, the service encapsulates the visualization and rendering techniques and merely accepts parameters for controlling the rendering instead of a specification of the rendering process.

The image-based approach offers advantages for styling (R7). A portrayal service that implements the mapping and rendering stages can offer control over these stages and, thus, a major part of the pipeline to a client. The offered types of styling are only limited by what can be expressed by output images. In contrast, a portrayal service that does not include the rendering stage leaves the responsibility for styling in the rendering stage to the client. This increases the complexity of the client and decreases interoperability. Additionally, the offered types of styling are restricted to what can be expressed by scene graph based representations.

In summary, the image-based approach directly supports meeting all the requirements identified in Section 2 except interactivity (R6). Efficiently supporting interactivity remains a major challenge that we address with the presented concepts in Section 5.

6.3 *Concepts for Image-Based, Interactive Visualization Clients*

In this subsection, we discuss how the three concepts for image-based, interactive visualization presented in Section 5 clients address the requirements identified in Section 2.

Advantages of the first presented concept based on additional service-side functionality (Section 5.1) include that clients can be exceedingly lightweight (R5), are easy to integrate with other applications (R1, R2), present original instead of approximated visual representations of the remote 3DGeoVE (R4), immediately display results of WVS requests with changed styling specifications in the next view (R7), and implement several interaction techniques efficiently by calling specific operations on the WVS instead of retrieving the needed source data and implementing the operations locally (R3). On the contrary, the concept offers no support for real-time navigation (R6), there is no reuse of image data between consecutive views (R3), and the efficiency and effectiveness of the implementation of specific interaction techniques depends on if and how well they are supported by specific service operations (R6).

Advantages of the second concept based on IBMR (Section 5.2) include that its implementation, hardware resource requirements, and integration efforts are only moderately complex (R1, R2, R5), it effectively provides low latency interaction and display updates (R6), supports several interaction techniques efficiently by exploiting the retrieved G-buffers from the WVS (R6), reuses the images retrieved from the WVS over several frames rendered locally on the client (R3), and displays results of WVS requests with changed styling specifications as soon as the limited set of locally maintained depth meshes and images

with previously requested styling are evicted from local memory (R7). On the other hand, aggregating locally a 3D reconstruction of visual representations of the remote 3DGeoVE based on depth meshes is not optimally effective and efficient (R1), retrieved images can only be reused for a limited number of frames rendered locally (R3), and the locally rendered novel views are approximations that suffer from hard to control under- and oversampling issues (R4).

In comparison to the IBMR concept, the advantages of the third concept based on PBMR (Section 5.3) include effectively providing low latency interaction and display updates (R6), more effective and efficient aggregation of the local 3D reconstruction from images (R1), reusing the images retrieved from the WVS for an extensive amount of time and numerous frames rendered locally on the client due to storing the 3D reconstruction out-of-core on local hard disk for latter reuse (R3), and the potential for locally rendering novel views with a higher visual quality and higher efficiency due to the improved aggregation and the multi-resolution data structure that exhibits less sampling issues (R4). On the contrary, the PBMR concept requires deleting the complete local 3D reconstruction when the styling specification for WVS requests changes since the 3D reconstruction inherently represents the results of the previously used styling specification (R7, R3). It requires more hardware resources than the IBMR concept due to the potentially large memory footprint of the local 3D reconstruction kept in memory and out-of-core on local hard disk and the computing intensive rendering of the 3D reconstruction (R5, R3). Moreover, creating an implementation for high quality, multi-resolution PBMR that efficiently utilizes potentially limited hardware resources and, in particular, a GPU, is complex and challenging (R5). Hence, implementation complexities, hardware resource requirements, and integration efforts of the PBMR concept are the most complex when comparing the three concepts (R1, R2, R5).

6.4 *Quantitative Results*

In the following, we present preliminary quantitative results of our initial, not yet optimized proof-of-concept implementations of a 3D rendering service implementing the WVS interface and three 3D clients implementing the three proposed concepts. Moreover, we report on initial industry impact of our work.

In the first experiment, we aim at measuring the rate at which the 3D rendering service can provide service consumers with rendered images. For this experiment, we created a service consumer that stresses the service by sending 40 requests to the service each requesting three image layers (color, depth, and object ID) for one 3D view. The service consumer sends up to 10 requests in parallel. The service processes each request sequentially. For each request, the service consumer receives one HTTP multipart response containing the three requested image layers. In total, the service generates and delivers 120 images. We measure the time for sending a request, rendering the images, compressing the images (JPEG, PNG), sending the images, and decompressing the images by the service consumer. The experiment is performed in an intranet environment. The service is executed on a desktop PC (Windows Server 2003, 1.86GHz double core, 2 GB RAM, nVidia GeForce GTX 260). The service consumer is executed on a different PC connected to the network. As a result, we measure that a service consumer can receive images at an average rate of 5.7 images per second for an image resolution of 512x512 and 2.6 for 1024x1024. In a second experiment, we measure the memory size of generated and transferred image layers while navigating through the 3DGeoVE. For an image resolution of 512x512, on average, color required 77.95 kbytes (JPEG), depth 199.63 kbytes (PNG), and object ID 9.56 kbytes (PNG). We expect to achieve higher delivery and compression rates in the future by applying advanced parallel processing and compression techniques.

In a third experiment, we measure the latency of interactions of a user with the 3D client implementation based on additional service-side functionality (Section 5.1). In summary, each interaction with the client that requires no requests from the WVS (e.g., defining marks, lines, and paths in the 3D view) shows no perceivable latency. The interactions that require requests from the WVS (e.g., moving the virtual camera, measuring paths) show latencies under 400 milliseconds. Requesting a new 3D view with the `GetView` operation (single color layer, 512x512) turns out to be the most time consuming operation.

In a fourth experiment, we measure the rendering rate of the 3D client implementation based on IBMR (Section 5.2). The implementation is based on Java and OpenGL. The client is executed in a web browser on a notebook (Windows XP, 2.4GHz double core, 3 GB RAM, nVidia Quadro FX 570M with 512 MB

RAM). In this experiment, we log the rendering rate of the client while a user navigates several minutes through the 3DGeoVE using different navigation techniques. While the user navigates, the client retrieves images (512x512 resolution) from the service as appropriate. The average rate of frames per second is zero when the user is not interacting with the client and the current view does not change since the 3D view is not updated in this situation, 284 when the user looks around from a fixed camera position (rendering images from the WVS organized locally as a single cube map), 102 when the user employs a fly navigation technique (rendering few depth meshes), and 71 when the user uses a goto navigation technique (rendering up to 12 depth meshes). We expect to achieve higher rendering rates in the future by applying adaptively triangulated depth meshes.

In a fifth experiment, we measure the rendering rate of the 3D client implementation based on PBMR (Section 5.3). The implementation is based on C++, OpenGL ES, and Apple iOS. The client is executed on an Apple iPhone 3GS. As in the previous experiment, we log the rendering rate of the client while a user navigates several minutes through the 3DGeoVE using different navigation techniques. While the user navigates, the client retrieves images (320x480 resolution) from the service as appropriate. In summary, the average frame rate is zero when the user is not interacting with the client and the current view does not change (as in the IBMR client), 20 when the user changes the virtual camera parameters (*interaction mode*), and below 5 for the duration that the virtual camera parameters stay constant (*quality mode*). In the interaction mode, the client can achieve a user defined frame rate (e.g., 20) by increasing the threshold for the projected screen space area of octree nodes and, thus, sacrifices visual quality for rendering speed. In the quality mode, the client sets the threshold to below one favoring quality over speed. In this mode, frame rates below one can occur despite the output-sensitive approach that already incorporates LOD. Reasons for this include that the implementation currently does not support occlusion culling (Akenine-Möller et al., 2008) (i.e., data is increasingly aggregated and used for rendering, however, occluded parts are not discarded early in the rendering), and that managing and rendering from a dynamic octree is generally not as efficient as when using a static octree.

The authors collaborated with an industry partner, Autodesk Inc., on work on the display client and the client based on IBMR. This collaboration lead to an integration of these approaches into products of our industry partner.

7 Conclusions

In this paper, we identified seven practically relevant requirements for 3D geovisualization systems with a focus on 3DGeoVE informed by existing literature. We introduced the fundamentals of the SOA paradigm, standards, and the distributed visualization pipeline. We presented a general approach for designing 3D geovisualization systems intended to meet the previously identified requirements. It was based on the introduced fundamental concepts, image-based representations, and the WVS. Three concepts for image-based, interactive visualization clients were presented as instances of the general approach. Finally, we discussed how the proposed general approach and the three concrete concepts support meeting the previously identified requirements.

As the key advantage of the image-based approach, the complexity that a client is exposed to for displaying a visual representation is reduced to a constant factor primarily depending on the image resolution. The client is shielded from the arbitrarily complex process and its data representations of generating visual representations from massive geodata. In summary, the image-based approach directly supports meeting all the requirements identified in Section 2 except interactivity. The three presented concepts exploit the complexity reduction of the image-based approach and already provide interactivity to different degrees. However, providing interactivity within a service-oriented, standards- and image-based approach remains a major challenge. Our future work aims at improving efficiency, the quality of visual representations, and providing interactivity on the same level as can be experienced in non-distributed 3D geovisualization systems.

Acknowledgements The authors thank Lars Schneider and Norman Holz for contributing to the implementation of the point-based rendering client and Autodesk Inc. for successful collaboration.

References

Open Geospatial Consortium (OGC) Website. URL, `http://www.opengeospatial.org/`, 2010. Accessed 31.3.2011.

Tomas Akenine-Möller, Eric Haines, and Natty Hoffman. *Real-Time Rendering*. A. K. Peters, Ltd., Natick, MA, USA, third edition, 2008. ISBN 987-1-56881-424-7.

Gennady Andrienko, Natalia Andrienko, Jason Dykes, David Mountain, Penny Noy, Mark Gahegan, Jonathan C. Roberts, Peter Rodgers, and Martin Theus. Creating Instruments for Ideation: Software Approaches to Geovisualization. In Dykes et al. (2005), pages 103–125.

Jens Basanow, Pascal Neis, Steffen Neubauer, Arne Schilling, and Alexander Alexander Zipf. *Towards 3D Spatial Data Infrastructures (3D-SDI) based on Open Standards - Experiences, Results and Future Issues*, pages 65–86. Lecture Notes in Geoinformation and Cartography. Springer, 2008.

Yaser A. Bishr. Overcoming the Semantic and Other Barriers to GIS Interoperability. *International Journal of Geographical Information Science*, 12(4):299–314, 1998.

K. W. Brodlie, J. Brooke, M. Chen, D. Chisnall, C. J. Hughes, Nigel W. John, M. W. Jones, M. Riding, N. Roard, M. Turner, and J. D. Wood. Adaptive Infrastructure for Visual Computing. In *Theory and Practice of Computer Graphics*, pages 147–156, 2007.

Ken Brodlie, David A. Duce, Julian R. Gallop, J. P. R. B. Walton, and Jason Wood. Distributed and Collaborative Visualization. *Computer Graphics Forum*, 23(2):223–251, 2004.

Chun-Fa Chang and Shyh-Haur Ger. Enhancing 3D Graphics on Mobile Devices by Image-Based Rendering. In *Proceedings of the Third IEEE PCM 2002*, London, UK, 2002. Springer-Verlag. ISBN 3-540-00262-6.

Jürgen Döllner. Geovisualization and Real-Time 3D Computer Graphics. In Jason Dykes, Alan M. MacEachren, and Menno-Jan Kraak, editors, *Exploring Geovisualization*. Elsevier Science, 2005.

Allan Doyle and Adrian Cuthbert. *Essential Model of Interactive Portrayal*. Open Geospatial Consortium Inc., November 1998.

Jason Dykes. Facilitating Interaction for Geovisualization. In Dykes et al. (2005), pages 265–291.

Jason Dykes, Alan M. MacEachren, and Menno-Jan Kraak, editors. *Exploring Geovisualization*. Elsevier Amsterdam, 2005.

Thomas Erl. *Service-Oriented Architecture: Concepts, Technology, and Design*. Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2005. ISBN 0-13-185858-0.

Daniel Filip. Introducing smart navigation in Street View. `http://google-latlong.blogspot.com/2009/06/introducing-smart-navigation-in-street.html`, 2009. Accessed 31.3.2011.

Jinghua Ge. *A Point-Based Remote Visualization Pipeline For Large-Scale Virtual Reality*. PhD thesis, University of Illinois at Chicago, 2007.

Enrico Gobbetti, Dave Kasik, and Sung-eui Yoon. Technical Strategies for Massive Model Visualization. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*. ACM, 2008. ISBN 978-1-60558-106-2.

Markus Gross and Hanspeter Pfister. *Point-Based Graphics*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007. ISBN 0123706041.

R.B. Haber and D. A. McNabb. Visualization Idioms: A Conceptual Model for Scientific Visualization Systems. In B. Shriver, G. M. Nielson, and L. Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93, Los Alamitos, 1990. IEEE Computer Society Press.

Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. Towards Advanced and Interactive Web Perspective View Services. In *Developments in 3D Geo-Information Sciences*. Springer, 2009.

Benjamin Hagedorn, Dieter Hildebrandt, and Jürgen Döllner. *Web View Service Discussion Paper, Version 0.6.0*. Open Geospatial Consortium Inc., February 2010.

Dieter Hildebrandt and Jürgen Döllner. Implementing 3D Geovisualization in Spatial Data Infrastructures: The Pros and Cons of 3D Portrayal Services. In Wolfgang Reinhardt, Antonio Krüger, and Manfred Ehlers, editors, *Geoinformatik 2009*, volume 35, pages 1–9. ifgiprints, April 2009. ISBN 978-3-89838-619-7.

Fabrizio Lamberti and Andrea Sanna. A Streaming-Based Solution for Remote Visualization of 3D Graphics on Mobile Devices. *IEEE Transactions on Visualization and Computer Graphics*, 13(2), 2007. ISSN 1077-2626.

Markus Lupp, editor. *Styled Layer Descriptor Profile of the Web Map Service Implementation Specification, Version 1.1.0*. Open Geospatial Consortium Inc., June 2007.

Alan M. MacEachren and Menno-Jan Kraak. Research Challenges in Geovisualization. *Cartography and Geographic Information Science*, 28(1):3–12, 2001.

Alan M. MacEachren, Mark Gahegan, William Pike, Isaac Brewer, Guoray Cai, Eugene Lengerich, and Frank Hardisty. Geovisualization for Knowledge Construction and Decision Support. *IEEE Computer Graphics and Applications*, 24(1):13–17, 2004. ISSN 0272-1716.

Steffen Neubauer and Alexander Zipf, editors. *3D-Symbology Encoding Discussion Draft, Version 0.0.1*. Open Geospatial Consortium Inc., 2009.

Michael P. Papazoglou, Paolo Traverso, Schahram Dustdar, and Frank Leymann. Service-Oriented Computing: State of the Art and Research Challenges. *Computer*, 40(11):38–45, 2007. ISSN 0018-9162.

Theresa-Marie Rhyne and Alan M. MacEachren. Visualizing Geospatial Data. In *SIGGRAPH 2004: ACM SIGGRAPH 2004 Course Notes*, page 31, New York, NY, USA, 2004. ACM.

Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. *SIGGRAPH Computer Graphics*, 24(4):197–206, 1990. ISSN 0097-8930.

Arne Schilling and Thomas H. Kolbe, editors. *Draft for Candidate OpenGIS Web 3D Service Interface Standard, Version 0.4.0*. Open Geospatial Consortium Inc., 2010.

Heung-Yeung Shum, Shing-Chow Chan, and Sing Bing Kang. *Image-Based Rendering*. Springer, 2007. ISBN 978-0387211138.

Robert Sisneros, Chad Jones, Jian Huang, Jinzhu Gao, Byung-Hoon Park, and Nagiza F. Samatova. A Multi-Level Cache Model for Run-Time Optimization of Remote Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):991–1003, 2007.

H. Wang, K. W. Brodlie, J. W. Handley, and J. D. Wood. Service-oriented approach to collaborative visualization. *Concurrency and Computation: Practice & Experience*, 20(11), 2008. ISSN 1532-0626.

Ji Soo Yi, Youn ah Kang, John Stasko, and Julie Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1224–1231, 2007. ISSN 1077-2626.