An Assisting, Constrained 3D Navigation Technique for Multiscale Virtual 3D City Models

Dieter Hildebrandt · Robert Timm

Received: 12 February 2013 / Revised: 14 July 2013 / Accepted: 14 August 2013

Abstract Virtual 3D city models serve as integration platforms for complex geospatial and georeferenced information and as medium for effective communication of spatial information. In order to explore these information spaces, navigation techniques for controlling the virtual camera are required to facilitate wayfinding and movement. However, navigation is not a trivial task and many available navigation techniques do not support users effectively and efficiently with their respective skills and tasks.

In this article, we presents an assisting, constrained navigation technique for multiscale virtual 3D city models that is based on three basic principles: users point to navigate, users are lead by suggestions, and the exploitation of semantic, multiscale, hierarchical structurings of city models. The technique particularly supports users with low navigation and virtual camera control skills but is also valuable for experienced users. It supports exploration, search, inspection, and presentation tasks, is easy to learn and use, supports orientation, is efficient, and yields effective view properties. In particular, the technique is suitable for interactive kiosks and mobile devices with a touch display and low computing resources and for use in mobile situations where users only have restricted resources for operating the application.

We demonstrate the validity of the proposed navigation technique by presenting an implementation and evaluation results. The implementation is based on service-oriented architectures, standards, and image-based representations and allows exploring massive virtual 3D city models particularly on mobile devices with limited computing resources. Results of a user study comparing the proposed navigation technique with standard techniques suggest that
the proposed technique provides the targeted properties, and that it is more advantageous to novice than to expert users.

**Keywords** Virtual 3D City Model · Multiscale Modeling · View Navigation · Virtual Camera Control · Mobile Device · Distributed 3D Geovisualization

1 Introduction

*Virtual 3D city models (V3DCMs)* serve as integration platforms for complex geospatial and georeferenced information and as medium for effective communication of spatial information. They can be provided to users as parts of *3D geovirtual environments (3DGeoVEs)*. In order to enable users to explore, analyze, and present these information spaces and to enable users to conduct wayfinding, moving and controlling the virtual camera within 3DGeoVEs, navigation techniques are required.

However, navigation in 3DGeoVEs is not a trivial task [48,18,24] and many navigation techniques do not support users effectively and efficiently with their respective skills and tasks in this specific type of virtual environment. Still the most commonly provided techniques in 3DGeoVEs and 3D computer graphics applications using 2-degree of freedom (DOF) devices include *pan*, *zoom*, *orbit*, *look*, *fly*, and *goto* [18,24]. These standard techniques exhibit several shortcomings that they share with many other techniques.

As a common issue, navigation techniques are not *easy to learn and use*. Navigation in 3DGeoVEs is a learned skill. Learning can be error-prone and confusing requiring significant effort even for experienced users [18,24]. Controlling simultaneously up to seven camera DOFs requires dexterity and is often found problematic [14]. Navigation techniques can be difficult to understand, provide insufficient information to the user, and may be ineffective in preventing users from making errors [18,14]. A core problem for users when navigating in 3DGeoVEs is *disorientation* [10,9,18,24]. It affects in particular inexperienced but also experienced users [18,9] and is a problem in particular in multiscale environments [31]. Moreover, navigation techniques can permit generating views of 3DGeoVEs with ineffective *view properties* [14] that can arise in the form of unsteady and discontinuous camera motion, awkward viewing angles presenting the model in poor light or missing important features, unwanted views, “chunky or visually jarring” views (e.g., due to frequent mouse clutching), distracting transitions, a low or inconsistent level of visual and interactive quality [38,25,10,9,11,14]. Furthermore, we can observe suboptimal *efficiency* of navigation techniques regarding human (e.g., time, physical efforts) and computing resources. Completing a single navigation task often requires a collection of navigation techniques resulting in frequent control switches, in particular when using standard navigation techniques [18,24]. Each control switch divides the action into separate chunks; leading to a higher separation in subtasks and a higher cognitive load, higher time consumption, inefficient movement trajectories, and ineffective view properties [38,10,11,24]. Many navigation techniques map input device DOFs to camera DOFs in *real-time* requiring
A 3D Navigation Technique for Multiscale Virtual 3D City Models

real-time 3D rendering, although this is not crucial for many applications using V3DCMs [17, 24]. In this case, alternative, selective navigation techniques can be more computer resource efficient [17].

To address these issues, in this article, we present an assisting, constrained navigation technique for multiscale V3DCMs that is based on three basic principles: users point to navigate, users are lead by suggestions, and the exploitation of semantic, multiscale, hierarchical structurings of city models (Fig. 1). The technique is intended to support particularly users with low navigation and virtual camera control skills but also experienced users. It is intended to support exploration, search, inspection, and presentation tasks, to be easy to learn and use, support orientation, to be efficient, and yield effective view properties. In particular, the technique is intended to be suitable for interactive kiosks and mobile devices with a touch display and low computing resources and for use in mobile situations where users only have restricted resources for operating the application. We demonstrate the validity of the proposed navigation technique by presenting an implementation and an evaluation including a user study. The implementation is based on service-oriented architectures, standards, and image-based representations (SSI) [27] and allows exploring massive V3DCMs particularly on mobile devices with limited computing resources.

This article first reviews related work (Section 2) and then provides the following contributions: a novel navigation technique for multiscale V3DCMs based on semantic navigation hierarchies, suggestions, and pointing (Section 3), an implementation of the navigation technique based on SSI and its integration into an existing 3D geovisualization system (3DGeoVS) (Section 4), an image-based representation of a navigation hierarchy (Section 4.1), an optimizing preprocessing pipeline and algorithms for the automated, generic generation of a navigation hierarchy from a V3DCM including an implementation (Section 4.2), an interactive runtime processing pipeline and algorithms using image-based representations provided by services as the sole representation of a 3DGeoVE (Section 4.3), an algorithm for the view-dependent, level-of-scale-dependent selection of nodes from an image-based navigation hierarchy (Section 4.3), a view overlay visualization technique structuring an independently generated view (Section 4.3), and an evaluation of effectiveness and efficiency aspects including a user study (Section 5). Section 6 concludes this article.

2 Related Work

Constrained navigation can be defined as “the restriction of viewpoint generation and rendering parameters to goal-driven subclasses whose access is specified by the application designer” [26]. It is a general principle that can be found in a multitude of approaches and is applied successfully to improve navigation [30, 9, 1]. Previous approaches have limited the camera’s position to viewpoints [47], 1D trajectories [38, 30, 17, 11], 1D trajectories connected through graphs [46, 1, 41], 2D surfaces [26, 10, 32], and 3D volumes [5]. Moreover, approaches imposed constraints on the camera’s velocity [38, 51] and orienta-
User control can be completely absent [15] or provided locally with varying degrees [26,10]. Constraints can essentially be authored manually [10, 11,1] or computed automatically [46,32]. This work constrains the camera to automatically computed 1D trajectories, velocities, and orientations connected through a graph with user control restricted to decisions at nodes. In contrast to previous work, the constraints are automatically computed from a semantic V3DCM with the intent to support specifically navigation in multiscale V3DCMs from the city level down to buildings without allowing explicitly to travel street networks.

**Automatic camera control** refers to the automated computation of static and dynamic camera parameters according to goal specifications typically without any direct user input [14]. Goals define the desired view properties and can be specified on (a) the camera parameters directly such as focal length, vantage angles, distances to objects [25], and collision avoidance [5], (b) camera path properties such as collision avoidance, path and frame coherence [25,13], and cinematographic properties [13], or (c) properties relating to the projected image content such as absolute [25] or relative locations of objects [42], framing of objects [42,4], occlusion [4,25], and properties relating to aesthetics and cognition [22] with canonical views [7] as one relevant concept. In this work, we apply established view properties such as vantage angles, collision avoidance, path and frame coherence, framing of objects, occlusion, and cognitive properties. However, we select, instance, and apply a specific subset of properties for the specific case of multiscale, hierarchical V3DCMs.

**Pointing** in this work refers to users indicating a single 2D position in screen space on the display by using a pointing input device. It is employed for navigation and camera motion in different ways such as to specify a direction, destination POI or path either **discretely** in time for selective navigation [17] in the current view [39], on a map or World in Miniature [46], using widgets [24], or **continuously** in the current view to specify real-time [38,40] or selective navigation, e.g., by sketching paths or gestures [17]. It is employed for specifying camera parameters [24] and to trigger complex actions [11,18]. Pointing can be used as the single, unified UI method [11] as can other simple methods such as dragging [10]. For specifying directions or destinations, other methods such as framing [39], gazing [40], or directly controlling camera parameters [52] can be used as well. In addition, continuous direct [45] or indirect [50] object manipulation techniques using one and more points including rotate-scale-translate interactions [45] can be used for indirect camera control. This work uses single-point pointing as the single, unified UI method to specify camera motion discretely. It differs from previous work in that it at the same time allows navigating in large-scale, multiscale V3DCMs, is used to specify directions, destinations, camera parameters, and complex actions, and does not require any authoring.

**Suggestions** in this work are referred to as navigation actions offered to the user for explicit, discrete selection in the presented 3D view (see Section 3.2). Suggestions can be represented as 3D models [44], 3D widgets [18], 2D thumbnails [11], 2D buttons [1], or text [29]. The actions triggered typi-
A 3D Navigation Technique for Multiscale Virtual 3D City Models

Critically include authored camera animations [11], computed camera animations between authored [44,1] or computed control points [15,18], “discontinuous” camera animation [44], and teleportation [47]. Suggestions triggering a camera to move from a current to a destination point in 3D space can also be called 3D hyperlinks [47,29] additionally differing in attributes such as duration, preview information, and visual continuity [47]. In this work, suggestions are represented as highlighted parts of a 3D model and 2D buttons triggering camera animations that are continuous, bidirectional, and computed completely automatically, have distance-dependent durations and minimized rotational offsets, have strong preview information, visual continuity, and directional cues, are integrated into a continuous Euclidean 3D space, and connect spatially adjacent and non-adjacent locations. Differing from previous work, we use such suggestions as the single, unified method to facilitate camera motion modeled with viewing graphs [20], provide suggestions to specifically support motion in large-scale, multiscale V3DCMs, compute and offer suggestions automatically, and provide a unique representation based on outlining and shading regions of the V3DCM in the view.

Semantic, hierarchical, multiscale modeling of geodata can be utilized to support navigation. The open standard CityGML [23] facilitates modeling V3DCMs with semantics and restricted hierarchical, multiscale characteristics limited to constituents and LOD representations of individual buildings. [17] exploit V3DCM single, small-scale semantics to interpret navigation commands. Hierarchies of visual representations of V3DCMs can be automatically generated representing the V3DCM with a range of levels of detail (LOD) and/or levels of abstraction (LOA) based on geometry and semantics (e.g., buildings, street networks) with the aim to improve perception (LOA) [12, 21] or rendering efficiency (LOD) [12]. Hierarchical structurings of navigable information spaces can be modeled with viewing graphs [20], and are employed to structure and guide 3D navigation [48,15,44,33,3]. Navigation in multiscale 3D models is found to cause disorientation potentially [31], is supported by authored hierarchies to model multiscale environments in the geospatial domain [44] and anatomy [33,3], and can be improved by offering adaptive or automatic control over movement velocity [38,51,53,39]. This work uses an automatically computed semantic, hierarchical, multiscale structuring of a V3DCM modeled using viewing graphs to support navigation. In contrast to previous work, the generated hierarchy structures the 2D city area, employs semantic features beyond buildings and streets such as administrative borders and geological features, and is generated and represented based on images. In addition, we use a semantic, multiscale hierarchy of a V3DCM at the same time to structure, guide, and constrain the navigation process, to automatically control movement velocity and view properties, and to support navigation with a specific visualization and navigation aids.

Image-based representations can be used to implement various functionalities of navigation techniques such as occlusion [25], collision [39,16], computing viewpoints with specific properties [49], or camera velocity control [51,39]. In these approaches, images are typically generated by a GPU and encode infor-
information such as geometry [39,16], object identification (OID) [25], or geographic attributes [54]. In contrast to previous work, we represent a hierarchical model as images and use this to support navigation: at preprocessing time as a stack of OID images and at runtime as view-dependent OID images encoding hierarchy in a unique manner. The navigation technique uses depth images as the sole representation of model geometry for computing camera animations, automatic framing (together with OID images), collision, and occlusion.

3 An Assisting, Constrained 3D Navigation Technique

In this Section, we describe the concept of the proposed navigation technique and its rationale. We introduce the technique with an overview, example, and characterization. Then, we present the concept in detail by explaining the three underlying principles, semantic, hierarchical, multiscale model (Section 3.1), navigation suggestions (Section 3.2), and point to navigate (Section 3.3), along with supplemental techniques (Section 3.4).

The proposed navigation technique (Fig. 1) allows the user to navigate in multiscale V3DCMs. For each view, it adaptively presents to the user a finite set of visual representations of navigation suggestions that represent specific potential user navigation intentions. The suggestions are derived from a semantic, hierarchical, multiscale structuring of a V3DCM. The user selects one of the presented suggestions by pointing at its representation. This triggers the technique to attempt to satisfy the intention represented by the suggestion by moving the camera continuously over time through 3D space to an appropriate target position. Once the target position is reached, this process starts again from the beginning. Figure 2 depicts an example navigation usage sequence descending from the highest to the lowest level of scale of a V3DCM.

The navigation technique can be characterized as selective [17], constrained [26,30,17], automatic [14], and assisting. In contrast to real-time navigation where the user controls the camera continuously over time using an input device, the user selectively, discretely issues navigation commands by pointing at visual representations of suggestions triggering camera movement to appropriate target positions. By applying constraints, we limit the camera movement to trajectories and orientations along these trajectories that are automatically computed to meet certain goals. We limit the users DOFs and, thus, where users can go with the intent to make it easier for them to get where they want to go [30]. The system assists users by automatically, adaptively presenting navigation suggestions.

3.1 Semantic, Hierarchical, Multiscale Model

The navigation technique utilizes a semantic, hierarchical, multiscale structuring of a V3DCM in order to support a user in navigating in a multiscale V3DCM. The structuring hierarchically subdivides a V3DCM by subdividing
Fig. 1: Screenshot of the 3D navigation technique displaying a 2D overlay on top of an independently created 3D view of a V3DCM. The overlay provides movement and view suggestions, navigation aids (overview map, compass, node and path labels) and divides the view into a focus and a context region with one node being selected.

its 2D ground area guided primarily by semantics (e.g., successively subdividing a city area into districts, quarters, etc., Fig. 3). The navigation technique uses such a hierarchy as a skeleton structure to guide the user’s navigation process and to constrain camera movement. For V3DCMs, various different types of hierarchies exist or can be conceived. Each is assumed suitable only for specific application domains and user tasks. For instance, for a tourism application, the V3DCM could be subdivided into districts, quarters, clusters of points of interest, and points of interest.

For a proof of concept case study, we use a type of hierarchy that is aimed at application domains such as city planning or city marketing. We are concerned with hierarchies that can be represented as rooted, directed, k-ary trees. Each node holds geometric (e.g., occupied 2D area), semantic (e.g., representing a district), and scale information (e.g., specifying scale of phenomena represented
Node geometries are partitions of their children’s geometries and are disjoint from each other on each level. We refer to each level $i$ of the tree as level of scale $i$ (LOS$_i$). The proposed hierarchy has seven LOS: city (root node), districts, quarters, boroughs, neighborhoods, blocks, and buildings. In Section 4.2, we demonstrate how to generate this type of hierarchy automatically from a given V3DCM. Figure 3 depicts an automatically generated, specific instance of this type of hierarchy for the city of Berlin.

Using a semantic, hierarchical, multiscale structuring to guide navigation in multiscale V3DCMs supports meeting the goals stated in Section 1 in several ways. First, research suggests that humans naturally organize their cognitive map into a semantic hierarchy [28]. Hence, using an explicit semantic hierarchy for visualization and navigation can help the user in correlating the presented V3DCM with his already existing cognitive map and in further developing his cognitive map. Second, carving up the V3DCM efficiently with large-scale semantics (e.g., dividing the city area into a set of named districts, etc.) yields a structure that users can read to find “good” paths to their desired target and, thus, supports wayfinding [20]. For this, a semantic structuring is assumed more effective than a structuring that is primarily guided by geometric criteria (e.g., quadtrees) [20, 44]. Third, a semantic, hierarchical structure scales well to large, multiscale environments and allows a user to traverse efficiently large models [20, 44]. A V3DCM can be represented with only seven LOSs (Section 4.2). The paths in the hierarchy between any two nodes tend to be short (e.g., for a balanced tree the maximum path length is logarithmic with the count of nodes). Fourth, we can use an explicit, discrete structuring of the V3DCM for visualization to structure explicitly and discretely the views of the V3DCM presented to the user. Such visualization has the potential to ease cognitive acquisition and processing. In addition, we can utilize the nodes of the structure
for selective navigation [17] as basis for stationary camera positions. Finally, geodata often contains emergent behavior at different scales [37]. A semantic, multiscale structuring can make the contained scale-dependent structures and information explicit and exploitable for visualization and navigation.

### 3.2 Navigation Suggestions

A navigation suggestion represents an executable navigation action that is offered by the navigation technique to the user for explicit, discrete selection in the presented 3D view. It realizes a specific potential user navigation intention for a specific stationary view. The navigation technique presents to the user for each stationary view a finite set of visual representations of automatically, adaptively computed navigation suggestions (Fig. 1). These are represented as 2D overlay elements displayed on top of a 2D projection of a V3DCM. The user can select a navigation suggestion by pointing at its visual representation triggering camera motion. We distinguish two types of navigation suggestions:

**Movement suggestions (MS)** represent the intention of a user to move the camera towards a part of the model as indicated by the visual representation of the MS. They are primarily intended to support search and exploration tasks. A MS is visually represented either by an outlined and shaded region...
Fig. 4: Augmenting the viewing graph at node $n$ with VSs of type yaw and pitch represented by the nodes $v_{ij} =: n_{v_{ij}}$ and connected edges (indicated by dashed lines). Each node $v_{ij}$ represents a combination of yaw $0 \leq i \leq n_{\text{yaw}}$ and pitch $0 \leq j \leq n_{\text{pitch}}$ with $n_{\text{yaw}} = 2$, $n_{\text{pitch}} = 3$ in this example.

of the displayed 2D projection of the V3DCM decorated with a textual label if available or by a GUI button.

**View suggestions (VS)** represent the intention of a user to acquire a different static or dynamic view on a currently focused region of the V3DCM. They are primarily intended to support object inspection tasks and are visually represented by GUI buttons.

We can model the navigable space that can be visited by using navigation suggestions and the provided views with their view properties with the concept of viewing graphs [20]. Each node holds information on geometry, semantics, scale, and camera 3D position and orientation. Each edge holds information on a camera position and orientation animation over time from the source to the target node. Thus, the viewing graph represents navigation suggestions as outgoing edges and view properties indirectly as camera parameters held in nodes and edges.

The basic navigation process using viewing graphs can be described as *view traversal* [20]. The camera is located at any given time either stationary at one node or is moving controlled by the navigation technique on the edge from a source to a target node. At any current node, the technique presents to the user a specific view of the node with a set of navigation suggestions corresponding to outgoing edges. By selecting a suggestion, the user triggers the camera to move automatically along the edge to the target node.

We construct a viewing graph from a given semantic, hierarchical, multiscale structuring (Section 3.1) by first adopting and then modifying its tree. Each node’s camera parameters are set up to look from the south at the node geometry’s center (*automatic north heading*), setting the pitch as a function
of the node’s LOS (from -90° at LOS₀ for 2D map-like views to -35° at LOS₆ for oblique views) (automatic pitch) and framing the node’s geometry’s 3D axis-aligned bounding box (AABB) (automatic framing). The automatic camera animations on the edges are set up with the goals to provide steady and continuous camera motion with high orientation values [9], the adequate level of visual information complexity required for supporting orientation, and rapid, controlled camera movement [38].

The navigation technique offers four types of VSs that are each implemented by adding nodes and/or edges to the viewing graph: yaw (view on focused region from indicated directions, e.g., 120° differences, “Look” button), pitch (views at pitches -90°, -35°, 0°, and pedestrian view, “Top”, “High”, “Mid”, and “Ped” buttons, Fig. 4), good view (automatic canonical views on focused region intended to be preferred by most users for a wide range of tasks[7]), and turntable (automatic 360° camera animation focused at region center with constant pitch and distance for overviews).

As additional MSs, we add to each node n additional edges to nodes visible from its view (Fig. 5) based on a projected screen space criterion (Section 4.3). As effects, the view is segmented into regions that tend to have equal sizes not falling below a threshold (Fig. 1), regions that are more distant are represented by nodes on lower LOSs (Fig. 6), visual clutter is reduced, the count and sizes of MSs in a view are controlled, and paths in the viewing graph are shortened. The intuition is that when a user points at a more distant location, he wants to move to a region around the location pointed at with a size that he can easily oversee from his current view instead of moving to that exact location. When the user selects a MS at node n to an adjacent node m, a target node is automatically chosen that has the smallest yaw and pitch difference to n, if n and m are spatially close. Otherwise, m is selected. Finally, the user can access the history of previously visited nodes (“Back” button) and move to a node’s parent region (“Up” button).

Navigation suggestions as presented support meeting the goals stated in Section 1 as follows. First, users can easily understand suggestions as a means to guide them. Suggestions are solely and consistently represented in two different, straightforward ways (outlining model regions and buttons) and selected in the same way always triggering camera movement. Suggestions can ease decision-making, one of the essential components of navigation. The navigation technique always presents to the user visually a clear, discrete, limited set

![Fig. 5: Augmenting the viewing graph at node n with additional edges (indicated by dashed lines) to hierarchy nodes as additional MSs.](image)
of available next actions. In particular, views that cannot be represented by directly outlining a region in the current view (e.g., same region from different direction), are made explicit and directly selectable as buttons without requiring complex manual steering. Structuring the views into semantic regions with edges can support perception, cognitive map building and wayfinding: The structures are comparable to Lynch’s [35] districts, nodes, paths, and edges, delineated by edges as important elements of views [36], and can act as morphological elements of the model that are found to largely define navigation behavior [2]. Second, user orientation is improved by providing error prevention through presenting for each view only a set of defined, assumable useful suggestions derived from the current view, error recovery (“Back” button), and continuous camera motion exclusively without any potentially disorienting teleportation [8]. Third, since the navigation technique takes control over possible camera positions, it can guarantee a certain level of quality and effectiveness for the view properties and never displays awkward views. Fourth, suggestions are efficient allowing rapid, controlled movement over long distances and selective navigation without requiring real-time rendering. Finally, the presented suggestions support exploration, search, inspection, and presentation tasks as well as wayfinding and motion. Furthermore, the suggestions conform to reported, observed user preferences: Users prefer manually tilted views at 3D objects only when they are more experienced, consistently return to overviews, return to north orientations after a short period of time, divide travel into motion and stationary phases for processing visual cues, and prefer to delegate navigation work to the system [2].
3.3 Point to Navigate

The navigation technique provides pointing as the single, unified user input method. Users point at suggestions provided by the technique to communicate their intentions and, thus, to issue navigation commands. Pointing refers to users indicating a single 2D position in screen space on the display by using a pointing input device (e.g., click with mouse) or a touch input device (e.g., one single finger tap). Pointing inputs are discrete in time and space. The mapping of input device parameters to camera parameters is not direct but instead decoupled and mediated by suggestions. Pointing at a suggestion represented by a button immediately triggers the action it represents. Pointing at an unselected, outlined region selects this region indicated by a blue coloring, pointing at a selected region triggers its action. In addition, the navigation technique provides automatic view magnification when the user points at or near a suggestion with a small screen space projection area \( a \) (i.e., \( a < \phi_{\text{min}} \), with \( \phi_{\text{min}} = 0.85 \text{ cm}^2 \) when using touch [43]).

Pointing supports meeting the goals stated in Section 1 as follows. First, pointing as an input method for navigation is easy to learn and use [38, 11, 24]. Users understand the concept of pointing to navigate very quickly [11, 24]. A single suggestion replaces and eliminates potentially complex motion and steering tasks. The navigation technique automatically, intelligently, and transparently performs and combines standard techniques (e.g., pan, zoom, and orbit to establish a view of an object) to control the camera DOFs. Thus, the user is relieved from the burden to control several techniques, switches between them and arbitrary navigation directly but instead controls navigation on a higher, more abstract level. Second, replacing direct camera control by higher-level suggestions transfers more control from the user to the navigation technique and, thus, enables it to improve user orientation and view property effectiveness. Moreover, pointing to navigate can be more efficient than standard techniques since for the same motion task it requires a smaller number of input events and less physical effort. Pointing is particularly suited for users with low camera control skills but is also useful for experienced users [24]. It conforms to the strong user expectation of being able to point at objects of the scene to trigger object-related actions [10, 11]. Finally, pointing only requires a generic pointing input device and no specific hardware.

3.4 Supplemental Techniques

Spatial information is provided by an auto-zooming 2D overview map with a fixed true-north orientation aligned with the camera’s automatic north heading (Fig. 1) and with an arrow indicating the camera’s position and direction. Hierarchical information is provided by a textual path label (indicating the hierarchy path from the root to the current node) and adaptive, hierarchical styling of the 2D overview map showing outlines of at most the three hierarchy
LOSs starting at the current node $n$ and highlighting via $FNC$ nodes on the path from $n$ to the root.

4 System Design and Implementation

In this Section, we present the system design and implementation of the proposed navigation technique. We provide a system architecture overview, introduce hierarchical object IDs as representation (Section 4.1), explain the automatic generation of a semantic, multiscale hierarchy for a V3DCM (Section 4.2), and the interactive, runtime processing pipeline (Section 4.3).

Figure 7 depicts an overview of a 3DGeoVS implementing the navigation concept introduced in Section 3. We use as basis an already existing 3DGeoVS based on SSI [27]. This system allows the interactive visualization of massive V3DCMs in particular on mobile devices with limited computing resources. We extend the system with support for the introduced navigation concept as follows. The Geodata Mapping service transforms V3DCM representations from the CityGML Storage service and the OSM Storage service into an image-based hierarchy and with this augments data in the Proprietary Model Storage. From the therein contained data, the 3D Rendering service generates 2D images of projected views of the V3DCM encoding various information such as depth or OID as requested. The 3D Visualization Client implements a 3D Client Rendering Technique and the 3D Navigation Technique that implements the proposed navigation technique on the client-side. The 3D Navigation Technique requests depth (encoding geometry) and OID (encoding hierarchy) images from the 3D Rendering service and additional hierarchy related data from the Thematic Data Storage service.
4.1 Hierarchical Object Identification (HOID)

The hierarchy is represented with hierarchical object identifications (HOIDs) and HOID images. An object identification (OID) is an integer value uniquely identifying an “object” or feature respectively within a 3DGeoVE. Let $T = (N, E)$ be a rooted, directed, k-ary tree with root node $r \in N$. Further, let $\text{lid}: N \setminus \{r\} \to N \setminus \{0\}$ map each node to an integer obtained by numbering for each tree node its child nodes locally with integers continuously starting from 1 with +1 increments, and $\text{bits}: N \to N, v \mapsto \lceil \log_2(v) \rceil + 1$ the minimum number of bits required to represent $v$. Then, we define $\text{hoid}: N \setminus \{r\} \to N \setminus \{0\}, n \mapsto \text{lid}(n_1) + \sum_{i=2}^{m} \text{lid}(n_i) \times 2^{\sum_{j=1}^{i-1} \text{bits}(\text{lid}(n_j))}$ with $P_n := (n_0 = r, n_1, \ldots, n_m = n)$ and $m \geq 1, \forall i, j, 0 \leq i < j \leq m: (n_i, n_j) \in E$ being a path from $r$ to $n$. Thus, $\text{hoid}$ maps each node $n \neq r$ to an OID whose value encodes node $n$ and the path from $r$ to $n$. A HOID is an OID calculated with function $\text{hoid}$. Though we limit HOIDs to encoding paths in rooted trees, we refer to this representation as hierarchical to stress this aspect. [19] proposes a similar encoding for quadtrees and octrees. Figure 8 depicts the allocation of bits in a HOID (32-bit integer) to the six levels of a specific hierarchy and a mapping of this integer to a RGBA color with the same size for use in images. Given a hierarchy $T = (N, E)$, a HOID image is a georeferenced 2D image of a projected view of a 3DGeoVE where each pixel encodes the HOID value of a node $n \in N$ whose geometry is projected to this pixel (Fig. 12).

4.2 Navigation Hierarchy Generation

In this Section, we describe the requirements for and design of a navigation hierarchy for a specific application domain and the implementation and service-based provisioning of its generation.

Case Study. As a case study, we create a navigation hierarchy for the domain of city planning and city marketing (Section 3.1). The hierarchy must allow traversing the complete city area from the city level down to individual buildings in a multiscale manner. We generate a hierarchy for the city of Berlin covering 892 km$^2$. Data from two different data sets is used, since neither contains complete sets of all required features: a CityGML data set [6] for the approximately 550,000 buildings and OpenStreetMap (OSM) for the remaining...
features. We use dimensions of 16,384\(^2\) pixel for representing the hierarchy and individual LOS as images with sufficient resolution.

**Requirements.** We identify requirements for the hierarchy that influence the effectiveness and efficiency of the navigation technique it is used in. When the camera is stationary at an inner node framing its geometry, all its children are visible and selectable in the view as MSs, covering on average 23\% of the screen space (Section 5.2). We suggest the children count to be in the range of \(\alpha_{\text{min}}=4\) and \(\alpha_{\text{max}}=40\) with a mean in the range of 14 ± 2 (\(\alpha_{\text{mean}}=14\), \(\alpha_{\text{mean}+\text{delta}}=2\)) for display sizes of minimum 7" diagonal and touch input. A minimum of 4 guarantees a nontrivial number of choices at each node and a sufficiently small number of steps from the root to a leaf and, thus, sufficient “movement speed”. A maximum of 40 yields average suggestion display areas on targeted displays that are above the minimum size that can be efficiently selected with touch input (0.85 cm\(^2\), [43]) and is assumed to yield still clear enough sets of choices. The targeted means of 14 ± 2 result from taking what the 7 ± 2 rule suggests [34] and doubling it in an attempt to follow the recommendation to prefer broad-shallow menu tree structures [34]. To allow for efficient touch-based selection, nodes offered for selection should have a minimum display size of \(\phi_{\text{min}}=0.85\text{ cm}^2\) [43] assuming approximately square node display projections. The hierarchy must be build from existing geometric properties of the V3DCM that are important in the targeted application domain and support user orientation. Hierarchy modifications, e.g., for enforcing child node properties as presented above, should preserve the original structure as much as possible.

**Hierarchy Design.** For the case study, a hierarchy with seven LOS can appropriately represent an approximately balanced tree with 550,000 building leaf nodes and \(\alpha_{\text{mean}}=14\). For the seven LOS, we select features as follows and, thus, create structures corresponding to Lynch’s [35] districts, nodes, paths, and edges; city (administrative boundaries), districts (ditto), quarters (ditto), boroughs (major streets and water bodies), neighborhoods (regular streets and railways), blocks (minor streets and ways), and buildings (building footprints). Each LOS forms a subdivision of its next higher LOS, if available, by combining its selected features with the features from its next higher LOS.
Every area enclosed by outlines from the selected features represents a node of the respective LOS.

Automated Generation: Processing and Algorithm. The Geodata Mapping service generates a navigation hierarchy from the V3DCM data and transforms it into a representation optimized for real-time 3D rendering. The service implements a generic, automated preprocessing pipeline as depicted in Figure 9. The service provides an interface based on the Web Processing Service (WPS) standard. It takes as input V3DCM data from the CityGML Storage and OSM Storage services, V3DCM triangle mesh data from the Proprietary Model Storage, and a set of configuration parameters (specifying data sources, LOS count, LOS feature selection rules, thresholds, etc.). The output is a set of multi-resolution terrain texture tiles and the input V3DCM triangle mesh augmented with vertex attributes.

The OSM LOS Visualization processing stage generates an Outline Image for each LOS from level 1 to 5 by filtering, mapping, rendering OSM City Model Data. Each generated Outline Image is a 2D image that contains the rasterized, top-down orthographic projections of all node geometries of a specific LOS. Similarly, the CityGML LOS Visualization stage generates an Outline Image for LOS containing buildings from CityGML Data. The ID Assignment, Artifacts Removal stage generates a LOS Image from each Outline Image by filling each outline area with a color representing an OID, removing outlines, and removing rasterization artifacts and very small features (Fig. 10).

![Fig. 10: LOS image artifacts removal. For each LOS image, unintended (a) small areas (yellow) and (b) slightly larger, oblong areas (yellow, middle), e.g., created by street lane outlines (left), are removed by growing their proximity (right).](image)

The properties of the hierarchy constructed up to this point depend primarily on the used data set and LOS feature selection rules. As will be evident (Section 5.2), the stated requirements (see above) are not met automatically. Therefore, we propose a greedy optimization algorithm that carefully modifies the hierarchy in order to meet better the stated requirements while preserving its essential structure. As preparation, the Tree Extraction stage extracts an explicit,
pointer-based tree data structure from the LOS images to allow for a more convenient and efficient implementation of the optimization algorithm. The Optimization stage takes as input the original LOS images and tree data structure, outputs optimized LOS images, and implements the algorithm presented in Listing 1 (procedure Optimize).

```
1 PROCEDURE Fit (parent, child)
2    geometryDiff = FindGeometryDiff(parent.geometry, child.geometry)
3    EnlargeGeometry(parent.geometry, geometryDiff.geometry)
4    ReduceGeometry(parent.los, parent, geometryDiff.geometry)
5
6 PROCEDURE Split(node, obb)
7    IF (NOT node.los.modifyingAllowed) || (node.size<σ.min)
8       RETURN
9    newNode = CreateNodeByOBBSplit(node, obb.shortestAxis)
10   IF (node.numChildren==1)
11       Split(node.children[0], obb)
12   FOR EACH child IN node.children
13       parent = FindParentWithMaxOverlap(child)
14       Link(parent, child)
15       Fit(child.parent, child)
16   PROCEDURE Optimize(treeNodeList)
17     DO
18        FOR EACH node IN treeNodeList
19           IF (node.numChildren >0) THEN
20              IF (node.numChildren >α.max) THEN
21                 Split(node, node.geometry.obb)
22              ELSE IF (node.numChildren <α.min) THEN
23                 child = FindChildWithMaxChildrenMaxSize(node)
24                 Split(child, child.geometry.obb)
25              ELSE
26                 child = FindChildWithMinArea(node)
27                 IF ((child.size/node.size)<τ.min) THEN
28                     Split(node, node.geometry.obb)
29                 END IF
30            END IF
31        END FOR
32        WHILE (SplitWasPerformedOnLastIteration())
33        WHILE (TreeChildrenCountMean() >α.mean+α.meandelta)
34           node = FindNodeSplittableFirstTimeMaxChildrenMaxSize()
35           IF (node==NULL)
36              BREAK
37           Split(node, node.geometry.obb)
38    END IF
39  END WHILE
40  END PROCEDURE Optimize
```

Listing 1: Pseudo-code of the greedy hierarchy optimization algorithm.

This algorithm modifies the tree only by splitting nodes. Splitting is more conservative than joining in the sense that it preserves all original geometry boundaries while only introducing new ones. Splitting a node \( n \) with parent \( p \) into \( c_0 \) and \( c_1 \) as described generally results in increasing \( p \)'s children count by one, on average an even distribution of the area and children of \( n \) among \( c_0 \) and \( c_1 \) (halving the children count), decreasing the tree's average children count, and aiming for OBB side length aspect ratios of one for \( c_0 \)'s and \( c_1 \)'s geometries. In this case study, the algorithm is configured to not allow splitting nodes on LOS\(_1\) and LOS\(_2\) (administrative entities) and LOS\(_6\) (buildings) since splitting a node on these LOSs would replace an entity directly defined in the input data set with clearly defined semantics (e.g., a certain district) by new entities with unclear, probably undefined semantics. In contrast, we allow splitting nodes on the other LOSs since they do not directly represent input entities but instead are defined by the outlines of such entities (e.g., streets). To allow for avoiding and limiting the number of child node splits while creating natural split border geometries aligned with the children geometries, we “fit” the two node geometries resulting from a split to their respective children’s geometries instead of splitting the children, if possible (Fig. 11). The algorithm
terminates eventually when no more splits can be performed because each node is either too small for splitting, is on a LOS that prohibits splitting, or has already been considered for splitting. To allow terminating the algorithm earlier and faster with an approximation of the end result, the algorithm can be extended to rate the importance of all known pending splits, process them according to importance, and terminate when the highest importance falls below a user-defined threshold. The algorithm has a complexity of $O(n)$ with $n$ being the tree’s node count. To extensively preserve the original structures and avoid excessive splitting, we relax the requirements for the mean children count and minimum area size by allowing means in the range $[4, 12]$ and smaller area sizes. We intend to compensate smaller area sizes with automatic view magnification (Section 3.3).

The Merge stage combines the set of optimized LOS images into a single HOID Image (Fig. 12). The Triangle Mesh Vertex Assignment stage assigns to each input building triangle mesh vertex an attribute representing the HOID of the building from the HOID Image. The Terrain Texture Generation stage removes LOSs from the HOID image and computes a tiled (e.g., $512^2$) image pyramid (mipmaps). We implemented a prototype of the preprocessing pipeline as an extension of the Geodata Mapping service in C++ using a single CPU core and no GPU as cross-platform software (operating on Linux, Mac OS X, Windows) in approximately 10k lines of codes (LOCs) reusing as notable third-party software PostgreSQL (as OSM data base), Mapnik (for OSM outline image rendering), and GDAL (for terrain texture generation).
4.3 Interactive Runtime Processing Pipeline

In this Section, we present the service-based implementation of the proposed \textit{3D Navigation Technique} as part of the \textit{3D Visualization Client}. The navigation technique implements an interactive, runtime processing pipeline as depicted in Figure 13.

Initially, the camera’s \textit{current view node} is set to the viewing graph node representing the root of the navigation hierarchy. The corresponding initial image requests and camera parameters are passed to the \textit{Image Retrieval} and \textit{Control Camera} processing stages respectively. The \textit{Image Retrieval} stage takes as input an image request specifying in particular camera parameters and image layers, and outputs the requested images. Typically, pairs of depth and HOID images of the same perspective projected view of a V3DCM are requested. They are retrieved according to availability either from the local image cache, the remote image cache of the \textit{3D Rendering} service, or are generated on demand by the \textit{3D Rendering} service.

The \textit{Suggestion Determination} stage derives the suggestions (Section 3.2) for the current view node from the HOID image and state information. The complete viewing graph is not represented explicitly but instead is created locally, partially, and on demand at each hierarchy node. To derive MSs represented as outlined regions in the view, we propose a \textit{view-dependent LOS selection} algorithm. Its purpose is to view-dependently select from the HOID image for the current view node the nodes that represent MSs and to prune the subtrees below the selected nodes. For the current view node, we determine the \textit{current hierarchy node} $n$ (e.g., node $n = v_{b1}$ in Fig. 4). We select nodes
that either are children of $n$ or are on the same or lower LOS than $n$ according to a hierarchical screen space criterion. The algorithm is structured in three phases: First, an explicit, pointer-based tree data structure is extracted from the HOID image for a more convenient and efficient processing. When the data structure is constructed, nodes above the LOS of $n$ except $n$'s children and subtrees below the children of $n$ are already pruned from the tree. Then, the tree data structure is recursed and a node is selected if it is either a leaf node or it is not on a path to $n$ and the mean projected screen space area of its children falls below a given threshold $\upsilon_{\text{min}}$. Finally, the HOID image is pruned by setting to zero in each HOID value the bits that represent nodes below selected nodes. Depending on the capabilities of the 3D Rendering service, the view-dependent LOS selection is performed either by the 3D Rendering service or locally by the client. Figure 14 demonstrates how the algorithm selects a set of nodes from the hierarchy represented view-dependently in the HOID image and how complexity is removed from a view. On average, the algorithm removes 98% of the nodes from a HOID image. Figure 6 illustrates how nodes from different LOSs are systematically selected for a given view.

The Suggestion Display stage displays the identified suggestions as a 2D overlay computed from the Pruned HOID Image and Button Specifications. To generate the outlined and shaded regions, a quad is positioned in 3D space to cover exactly the screen when the viewpoints of the camera and HOID image match. It is rendered with a fragment shader that takes as primary input the pruned HOID image, the current hierarchy node’s HOID, and the user-selected node’s HOID. The shader detects and draws edges between different nodes identified by their HOIDs, colorizes the user-selected node, and applies a FNC vignetting technique by increasing the brightness of the current hierarchy node’s children as the focus and decreasing the brightness of all other nodes as context. Textual node labels are displayed centered at projected node centers when available in the data once retrieved at startup from the Thematic Data Storage service.

In the Suggestion Selection Stage, the user selects a suggestion by pointing at its visual representation that is then executed in the Suggestion Execution Stage. Automatic view magnification is implemented by animating the camera’s FOV and orientation. Selecting a suggestion triggers a camera animation for which the positions and orientations of the target view node and trajectory must be
Fig. 14: View-dependent level of scale selection. From an example original view (top, left) and HOID image (top, right), nodes are removed ($v_{\text{min}}=1.85\,\text{cm}^2$) resulting in a less complex view (bottom, left) and pruned HOID image (bottom, right).

computed. For this, information on the environment is solely retrieved from pairs of depth and HOID images of three different view types used depending on the suggestion type: For MSs, current perspective views (retrieved for each current view node) or global overviews (for “Up”; top-down orthographic projection of the entire city area; retrieved once; e.g., $1024^2$ resolution) are used. For VSs, current perspective views or local overviews (top-down orthographically projected views enclosing all VS nodes of the current hierarchy node; retrieved for current hierarchy nodes; e.g., $32^2$ resolution) are used according to availability. Local and global overviews are retrieved additionally to provide information on small and large scales that may not be contained in the current view due to occlusion or the limited FOV. For computing the camera trajectory to the target node, its AABB is computed from the respective depth and HOID images by back-projecting depth samples identified by HOID values and building the AABB from them, the camera’s target position and orientation is computed relative to the AABB, and approximate occlusion management and collision avoidance is performed based on the assumption of a 2.5D V3DCM by projecting 3D lines of sight and movement into respective depth images. Once the camera parameters are computed, the camera animation is started and the images for the target view node are requested. We implemented a cross-platform prototype of the runtime pipeline as an extension of the 3D Visualization Client in C++ for tablet computers and smartphones (iOS) and PCs (Linux, Mac OS X,
5 Evaluation

We assess to what degree the proposed approach meets goals and requirements stated in Sections 1 and 4.2. For all experiments, we use as a case study the V3DCM of Berlin (Section 4.2) and as algorithm parameters $\alpha_{min}=4$, $\alpha_{max}=40$, $\alpha_{mean}=14$, $\alpha_{meandelta}=2$, $\sigma_{min}=20$ pixel, $\tau_{min}=0.001$, $\nu_{min}=1.85 \text{ cm}^2$, $\phi_{min}=0.85 \text{ cm}^2$, $\psi_{min}=0.92 \text{ cm}$. For the server, we use a workstation with an Intel Xeon W3520, 2.67 GHz, 4 cores, 12 GiB main memory, GeForce GTX 275, Seagate ST32000641AS, Windows 7 64-bit. For the clients, we use an Apple iPad2 and a workstation with the same specification as the server computer.

5.1 Computer Resource Efficiency

Figure 15 presents the preprocessing pipeline mean execution times of the complete process (66:08 minutes) and its stages. We estimate that the execution times can be considerably reduced by optimizing our prototype implementation and by exploiting parallelism. Main memory requirements primarily depend on the count, dimensions, and OID ranges of LOS images and are below 3.0 GiB for the case study.

Figure 16 depicts the mean execution times of the four most time consuming parts of the runtime processing pipeline on three different client hardware configurations. The client requires at any time main memory for one or two perspective views, zero to two local overviews, and one global overview, and memory and/or disk space for an image cache of arbitrary size. The client retrieves pairs of depth and HOID images from the 3D Rendering service. The requested image dimensions depend on the view type and screen dimensions. Table 1 reports mean network image transmission sizes for requested image dimensions. Executing LOS selection on the service-side relieves the client from its most time consuming processing stage and reduces the network transmission volume for perspective view HOID images by 90%.
Fig. 16: Mean execution times of the four most time consuming parts of the runtime processing pipeline on three different client configurations.

Table 1: Mean network image transmission sizes (byte).

5.2 Suggestion Count and Size

The optimization algorithm significantly improves the properties of the initial hierarchy (Tab. 2) towards the targeted properties (Section 4.2). However, achieving the targeted properties cannot be guaranteed for every input hierarchy since required node splits could not be performed in every case. At runtime, the LOS selection algorithm (Section 4.3) removes on average 98% of the nodes (9.7" display at 1024x768). Inside the focus, mean node display sizes (4.6 cm$^2 \geq \phi_{\min}$) and mean node counts (12 in $c_{\text{mean}} \pm \alpha_{\text{meandelta}}$) are in the targeted ranges, whereas minimum node display sizes and node counts are outside the targeted ranges due to runtime factors (e.g., occlusion, perspective shortening, framing, viewport clipping).

5.3 User Study

Goals and Hypotheses. In order to evaluate the general usability of the navigation technique, we conducted a user study. We evaluate to what extent the proposed navigation technique (HSP, for hierarchy, suggestions, pointing) meets requirements identified in Section 1 for novice and experienced users in comparison to standard navigation techniques (STD). We summarize our research hypotheses as follows: There is a difference when using STD or HSP for
Table 2: Navigation hierarchy node counts for the seven LOSs and the overall hierarchy before (left) and after (right) the preprocessing optimization stage.

Before Optimization | Final, after Optimization
<table>
<thead>
<tr>
<th>Level</th>
<th>Semantics</th>
<th>Node Count</th>
<th>Children Count</th>
<th>Node Count</th>
<th>Children Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>City</td>
<td>1</td>
<td>12</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>1</td>
<td>District</td>
<td>12</td>
<td>2</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Quarter</td>
<td>96</td>
<td>20</td>
<td>96</td>
<td>20</td>
</tr>
<tr>
<td>3</td>
<td>Borough</td>
<td>708</td>
<td>12</td>
<td>772</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Neighborhood</td>
<td>1,357</td>
<td>146</td>
<td>3,450</td>
<td>40</td>
</tr>
<tr>
<td>5</td>
<td>Block</td>
<td>13,199</td>
<td>2,231</td>
<td>31,794</td>
<td>39</td>
</tr>
<tr>
<td>6</td>
<td>Building</td>
<td>541,871</td>
<td>-</td>
<td>541,871</td>
<td>-</td>
</tr>
<tr>
<td>0-6</td>
<td>Tree Overall</td>
<td>557,244</td>
<td>2,231</td>
<td>577,996</td>
<td>40</td>
</tr>
</tbody>
</table>

Table 2: Navigation hierarchy node counts for the seven LOSs and the overall hierarchy before (left) and after (right) the preprocessing optimization stage.

Experimental Design. We adopt a 2x2 split-plot design with two independent variables: The navigation technique (STD or HSP) is the within-subject factor and subject experience with 3D interaction (novice or expert) is the between-group factor. Table 3 lists the assessed 9 dependent criteria (first column) and associated 14 dependent variables with their metrics (second column). The 3 variables for interaction efficiency in the search/inspection task are measured as finger tap count, direct manipulation time (total time at least one finger touches the display), and task completion time. All other 11 variables are measured based on subjective ratings using a questionnaire (questions listed in Table 3, second column) employing a 7-point Likert scale ranging from 1 (strongly disagree) to 7 (strongly agree), presented as an approximate to an interval-scale. Subjects performed three different types of tasks each in a given, delimited area of the V3DCM: In the first task type, a subject observed a presentation given by an experimenter resembling a virtual city tour visiting 15 sites consecutively. The second task type comprises exploring the V3DCM for 5 minutes and then as motivation to acquire spatial knowledge drawing a map of major structural elements of the V3DCM and notable spots or landmarks. In the third task type, a subject searches the V3DCM for 15 buildings each marked with two signs on its facade: a large sign and a small sign each showing two letters. Typically, after a large sign is discovered from a distance, the subject moves closer to the building and performs a building inspection to discover the small sign. For each building, the subject notes the letters from both signs on paper.

Apparatus. We use an iPad2 mounted on a table, inclined towards a sitting user. In order to eliminate network access latencies, all data required by the system is precached on the device. For use in the trials, we identified two disjoint areas in the V3DCM of Berlin each covering LOS2 to LOS6 with similar ground area and complexity. For STD, pan (one-finger drag), orbit (two-finger drag), zoom (two-finger pinch), and goto (double-tap) are offered. In addition, an
Table 3: Quantitative results of the user study. Evaluation criteria are refined into dependent variables and metrics. For each dependent variable, a mean value for measurements (7-point Likert scale for questions) is displayed for each of the four conditions \{Novice, Expert\} × \{STD, HSP\}. Results of significance tests for comparing pairs of dependent variable means are reported at $p<.05$ ($z$: Wilcoxon signed-rank test, $t$: dependent $t$-test, $U$: Mann-Whitney test, $s$: significant, $ns$: not significant). Combinations of a test result and the comparison of two means are illustrated with color-coding (green: $s$ and right better than left, yellow: $ns$, red: $s$ and left better than right).

overview map (Section 3.4) is provided with the restriction to outline only the extent of the current task's area.

Participants. Twenty subjects participate in the experiment. Ten subjects (four female, aged 25 to 46) are classified as novices (not using interactive 3D applications in a typical week), the remaining ten (three female, aged 24 to 39) as experts (daily or weekly use). All subjects are right-handed, hold at least one university degree, and rate their degree of being familiar with the two used city areas below 3 on a 7-point Likert scale.

Procedure. After a subject is welcomed and introduced, the subject first performs the presentation task using technique A and area A, and then using technique B and area B. Then, technique A is explained to the subject followed by a practice session, in which the subject is explicitly asked to adjust the technique's animation speed or sensitivity to fit personal preferences. With technique A in area A, the subject first performs the exploration task followed by a practice run and an actual run of the combined search/inspection task. Then, the procedure of introducing a technique and performing the exploration and search/inspection tasks is repeated with technique B and area B. After completing all tasks, the subject fills out a short questionnaire and receives a small gratification. Subjects are given a break after finishing each task. We randomize the assignment of technique STD and HSP to A and B, the two available areas to area A and B, the subject order, and sets of sites and buildings for the exploration and search/inspection tasks.

Results. Table 3 and Figure 17 present the quantitative experiment results and can be summarized as follows. Regarding subjective dependent variables
measured with questions Q1 to Q11 using a 7-point Likert scale, statistically significant results suggest that for novices (Tab. 3, column Novice) HSP in comparison to STD is easier to learn and use, allows better orientation, yields more effective view properties, gives the impression of higher navigation speed, is more satisfactory, and is preferred over STD as a default technique for V3DCMs. However, novices feel navigating less freely with HSP than with STD. On the contrary, results indicate that experts do not profit from HSP as much as novices do. For experts (column Expert), HSP in comparison to STD is easier to use, and yields more effective view properties when observing a presentation. However, experts feel navigating less fast and less freely with HSP than with STD. In all remaining dependent variables, no significant difference between HSP and STD is found. When comparing novices and experts using STD (column STD), experts rate higher all dependent variables except view properties when observing a presentation, speed of movement and general preference for V3DCMs. When using HSP (column HSP), there is no significant difference between novices and experts concerning ease, orientation, view properties, direct manipulation time, subjective precision, and general satisfaction. However, novices feel navigating faster and more freely and prefer HSP as default technique stronger than experts do, whereas experts value HSP higher for specific tasks. Regarding objective, efficiency-related dependent variables, statistically significant results suggest that for both novices and experts HSP in comparison to STD is more efficient regarding physical demands. However, no significant difference in task completion time is found for both novices and experts for HSP compared to STD. When comparing novices and experts, experts performed significantly more efficient when using either STD or HSP regarding physical demand and task completion time.
5.4 Limitations

The usability of the navigation technique largely depends on how well the provided suggestions match the user’s requirements. They can be insufficient in several regards. Suggestions for required views could be missing, e.g., when certain features are only visible in narrow ranges of yaw and pitch values and viewing these features is required but not covered by provided VSs. To mitigate this, suitable VSs could be computed adaptively for each feature based on a feature analysis and the navigation technique could be augmented with constrained (e.g., 1D path or 2D surface) or unconstrained direct camera control techniques to allow for more precise control. Features required as navigation targets could not be represented as suggestions. They could generally not be represented as suggestions (such as streets in the case study) hindering or prohibiting to focus or inspect these features. To mitigate this, MSs could be extended to node boundary edges and intersections for improved inspection of paths, edges, and nodes [35]. Alternatively, targeted features could be represented as suggestions that are not selectable at the current view but are clearly visible (such as a large building visible from a great distance). Instead of directly traveling to the feature, this could result in traveling through multiple intermediate nodes and visually following the target feature “jumping” to a different screen location at each node. To mitigate this, the navigation technique could be augmented with framed zooming [39] or direct camera control techniques. Required scales could be missing. The case study is limited to a scale range from city to building level and does not include indoor or pedestrian navigation. To support indoor or pedestrian navigation, the viewing graph could be extended with a ground-level routing network with edges along walkable linear structures (e.g., streets, corridors) and nodes at junctions.

Furthermore, the navigation technique only decorates an independently generated view with an overlay (de-) emphasizing features. Generating views and appropriately representing multiscale information from the V3DCM at various scales is an orthogonal problem that the navigation technique does not address. Additionally, given V3DCM data could result in suboptimal navigation hierarchies even after optimization (Section 4.2). Finally, since the hierarchy generation is parametrized by the target display size and input device type, changing these parameters considerably could require rebuilding the hierarchy.

6 Conclusions

In this article, we introduced the concept, implementation, and evaluation of a novel assisting, constrained navigation technique for multiscale V3DCMs based on semantic navigation hierarchies, suggestions, and pointing. We demonstrated how a navigation technique can be built based on SSI that supports novice and expert users in exploration, search, inspection, and presentation tasks, and that is easy to learn and use, improves orientation, is efficient, and yields effective view properties. Results of a user study suggest that the proposed navigation
A 3D Navigation Technique for Multiscale Virtual 3D City Models

The proposed navigation technique provides the targeted usability properties with the drawback of subjectively moving less fast and freely compared to standard techniques, and that it is more advantageous to novice than to expert users. The proposed navigation technique is not assumed to replace or generally be more advantageous than standard techniques. Instead, it can be seen as an alternative that could be more appropriate when facing specific requirements such as targeting novice users at interactive kiosks or on mobile devices in 3DGeoVSs built on SSI.

In future work, the approach can be further extended and evaluated by overcoming current limitations as discussed in Section 5.4, applying the overlay structuring visualization technique demonstrated for SSI and selective navigation to real-time rendering and standard navigation techniques, evaluating different types of semantic hierarchies and different V3DCMs, and conducting further user studies to more extensively assess the proposed navigation technique.

Acknowledgements The authors thank Jürgen Döllner for support.

References