Computers, Environment and Urban Systems 54 (2015) 95–107

Contents lists available at ScienceDirect

Computers, Environment and Urban Systems

journal homepage: www.elsevier.com/locate/ceus

# Semantic volume texture for virtual city building model visualisation

Lin Li [a,b], XinQiao Duan [a], HaiHong Zhu [a,*], RenZhong Guo [c], Shen Ying [a]

[a] School of Resource and Environment Science, Wuhan University, 129 Luoyu Rd., Wuhan, 430079, China
[b] Geo Spatial Information Science Collaborative Innovation Center of Wuhan University, 129 Luoyu Rd., Wuhan, China
[c] Urban Planning, Land and Resources Commission of Shenzhen Municipality, 518034 Shenzhen, China

ARTICLE INFO

*Article history:*
Received 19 September 2014
Received in revised form 19 July 2015
Accepted 19 July 2015
Available online xxxx

*Keywords:*
CityGML building model
3D cadastre
Illustrative visualization
Visual exploration
Semantic volume texture

ABSTRACT

With the rapid urbanisation and development of three-dimensional (3D) space use, space objects in residential houses are of increasing concern. Illustrating these spatial entities within a clustered and multi-layered environment is confronted with the long-standing cognitive challenges of visual occlusion, visual clutter and visual navigation. Direct illustration as cross sections and cutaways provide valuable instruction for removing occlusion while preserving global contexts to give positioning cues. However, cross-sectioning or cutting away of the popular boundary-described models suffers from computational robustness and efficiency problems, while separated boundary geometry with surface properties prevents the efficient image-based direct illustrations from implementing a visually complete, semantically consistent practice easily. This article proposes a semantic volume texture (SVT) model for direct illustration. This true-3D raster model integrates spatial pattern embedding as well, thus avoiding the costly amendments to keep semantic consistent and visually complete during illustrative cutting and reconstructing operations. The proposed model is extended to the practical base of CityGML schema, the preparation of SVT is presented and applications imitating cross sections and cutaways are demonstrated. Experiments show that SVT-based direct illustrations are effective and efficient, making the proposed model suitable for explorative visualisations in the layered micro-scale environments.



## 1. Introduction

Ever since industrialisation era, houses have been crowded together into the narrow streets with the urbanisation process. At the same time, the space within a house has been divided into component parts according to functional, structural and legal requirements. It gradually makes the semantic meaningful while geometric complicated layered virtual building environment. These micro-scale environments concern most people with important aspects from everyday indoor activities to proprietary entitlement. The necessity for visual styling and exhibiting the special-interest objects from hidden positions with clear shape, distinguishable appearance and essential spatial relationship to surrounding objects has also emerged and been increased ever since. Among the plethora of illustrative approaches are the cross sections (Fig. 1), cutaways (Fig. 2) and notable industrial artworks of Frank Soltesz.

Nowadays, space usage stretches vertically above and below the land surface. The increased demand for residential space promotes a more flexible illustrative visualisation for inspecting, managing and planning purposes on the refined virtual building models. As a result, there are more demands on interactive illustration or visual exploration of the indoor environments (Hong et al., 2015; Isikdag, Zlatanova, & Underwood, 2013; Goetz, 2012) and three-dimensional (3D) cadastre (van Oosterom, 2013) models, rather than static manual illustration artworks (Zhou, Dao, Thill, & Delmelle, 2015; Shojaei, Kalantari, Bishop, Rajabifard, & Aien, 2013; Wang, Pouliot, & Hubert, 2012; Mao, Harrie, & Ban, 2012; Zhu et al., 2011). However, direct illustrative exploration as cross sections or cutaways of the layered building environment is confronted with some fundamental challenges.

For most 3D building models using boundary description, the first challenge might come from the visual style mapping conflicts. Taking CityGML for example, thematic surface textures and materials are attached to the boundary geometry. When mapping these surficial properties on the topologically shared facets, there will be conflicts. CityGML primarily adopts the *generic coverage scheme* (ISO-19123) to avoid this conflict in the *appearance model* (Fig. 3), which presumes the presence of sufficient observable surfaces. It can be seen that the coverage scheme mainly focuses on landscape scenarios (Open Geospatial Consortium (OGC), 2012), and does not guarantee a complete and consistent visual description of the target feature. This kind of inherent deficiency of describing objects that are hidden by outer layers implies that the appearance scheme may need to be extended for exploration tasks where focus targets are switched dynamically.

Vector boundary modelling with separated surface properties brings severe computational challenges to cross-section and cutaway illustrations. Xie, Zhu, Du, Xu, and Zhang (2013) recently surveyed the

* Corresponding author.
*E-mail address:* hhzhu@whu.edu.cn (H. Zhu).

http://dx.doi.org/10.1016/j.compenvurbsys.2015.07.003
0198-9715/

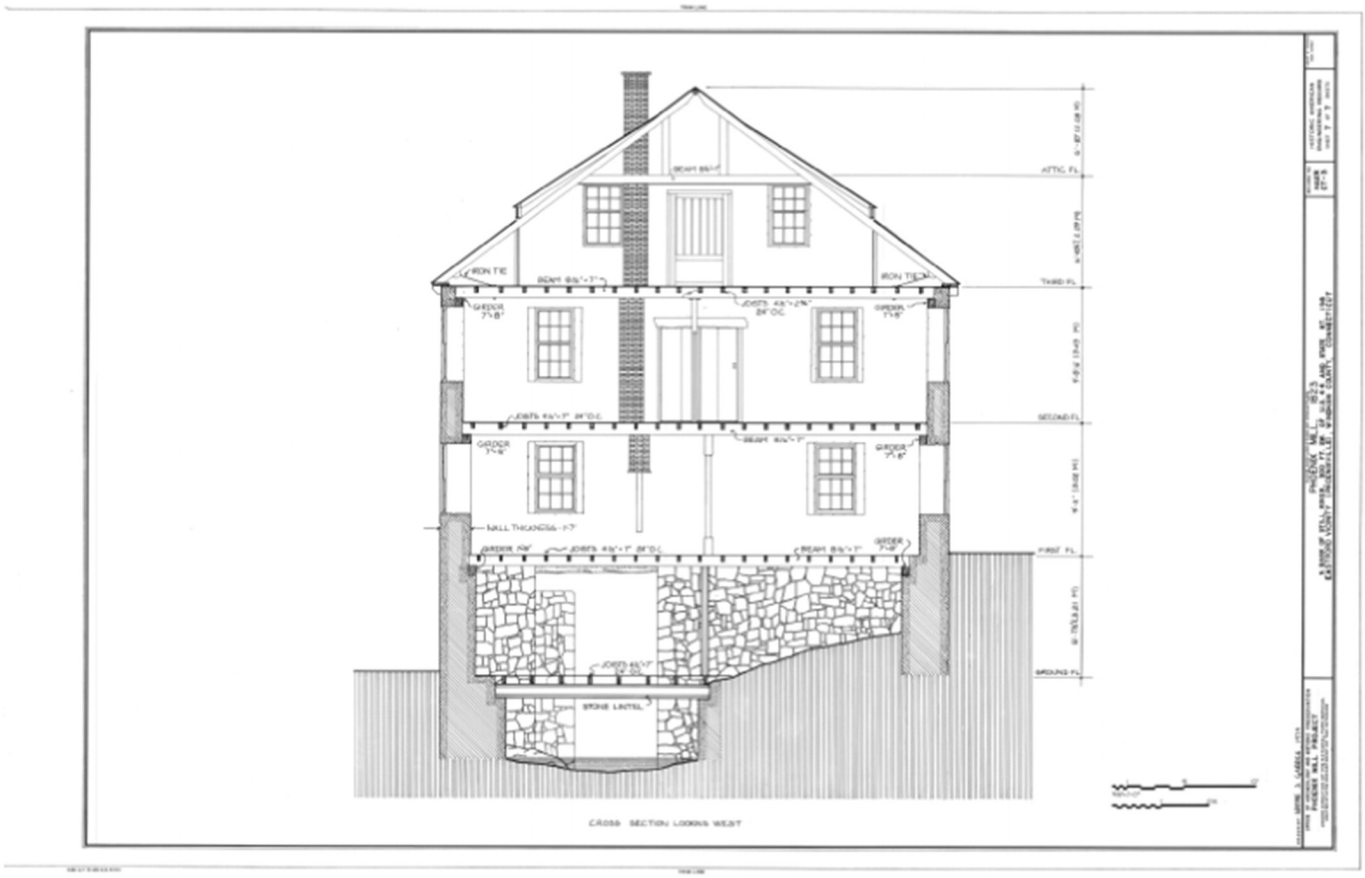


**Fig. 1.** Cross-section diagram of an old building in the United States. Cross section looking west – Phoenix Mill, North bank of Still River, Phoenixville, Windham County, CT, USA. HAER Register Number: HAER CONN,8-PHOE,1 (sheet 7 of 7).
From Library of Congress, Prints and Photographs Division, Washington, D.C. 20540 USA, http://hdl.loc.gov/loc.pnp/pp.print. HAER (Historic American Engineering Record) (1974).

difficulties of cutting polygonal meshes with and without borders (open surface objects and solid objects, respectively). Typical algorithms employ expensive numerical approaches, and they do not always warrant a topology correct cutting result (Rossignac, Megahed, & Schneider, 1992). After geometry cutting, there should always be a closure operation to make the cut objects 'solid', along with a high-cost joint-updating operation that makes geometry, topology and semantic consistent in order to ensure that the cut objects remain positively distinguishable (Burns & Finkelstein, 2008; Trapp & Döllner, 2013; Xie et al., 2013). Issues of robustness and high computational cost hinder the applicability of direct cross-sectioning illustrations on the boundary description model.

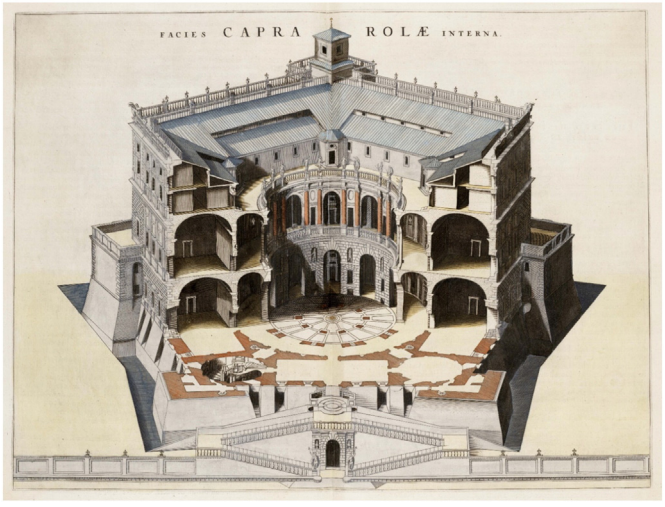


**Fig. 2.** Cutaway view of Villa Caprarola. Cutaway illustration differs from cross section in that it uses no single cut-plane and provides a '3D impression' rather than a planar diagram.
From "Theatricum Civitatum et Admirandorum Italiae Amsterdam", by Johannes Blaeu 1663.
Image from Auctionata online©.

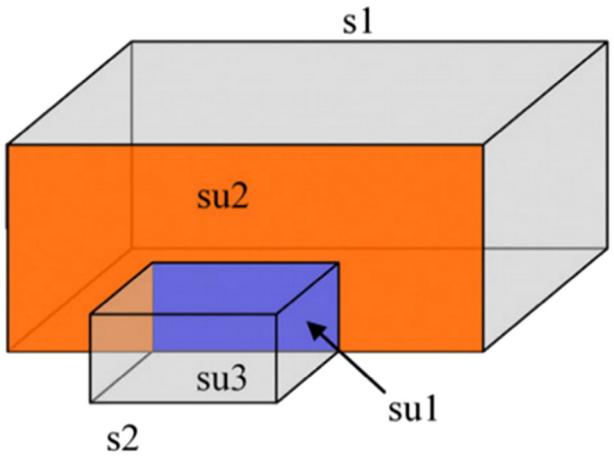


**Fig. 3.** Coverage scheme of CityGML. A building solid *s1* and a garage solid *s2* topologically touch at facet *su1*; the facet *su2* of solid *s1* is divided into two parts, the shared part *su1* will contribute no 'surface data' to *s1*'s appearance. In other words, when solid *s1* is singled out for illustration, it will have no complete surface description data.
Image from Gröger & Plümer (2012).

The aim of cutaways is to directly remove visual occlusion, while illustrative insights behind imply a focus + context paradigm (Diepstraten, Weiskopf, & Ertl, 2003; Ropinski & Hinrichs, 2005; Kruger, Schneider, & Westermann, 2006; Knödel, Hachet, & Guitton, 2009). The challenges here concern not only computations, but also some profound cognitive issues. The first challenge is the extraction of target object or region of interest (ROI) through cutting or spatial Boolean operations; and there should also be a consistent visual styling on it for observing from any outside position. Second, direct cutting-off faces the problem of how deep and to what extent the occluding layers (or objects) should be cut. For this reason, a viewpoint with target defined cutting is the more appreciated choice. As the cut-off area (view area) is usually larger than the target hull, there should also be a closure operation and joint-updating operation on the surrounding *clip-surface*(*s*). Third, we must condense the contextual information about the target by data reduction, model abstraction or generalisation. Contextual information provides essential positioning cues, but too much nearby detail would overwhelm the observer. Finally, we wish to composite the highlighted target with suppressed surrounding context using multi-pass rendering synthesis. It can be seen that, besides the ambiguous questions of cognition (styling, view area definition and generalisation), computational problems concerning remeshing and closure are similar to those encountered in cross sections, but to a much greater degree. In addition, the composition process commonly involves dynamic visibility computation of multiple occlusions (Knödel et al., 2009; Trapp, Beesk, Pasewaldt, & Döllner, 2011). The dynamic visibility computation in a complex scene would prevent the rendering algorithm from achieving a satisfactory frame rate on the basis of the object-order algorithm, which is commonly adopted by boundary representations (Gross & Pfister, 2007; Knödel et al., 2009).

Because of the aforementioned fundamental challenges, many direct illustrations on polygonal scenes turn to diverse image-based techniques to improve performance (Diepstraten et al., 2003; Li, Ritter, Agrawala, Curless, & Salesin, 2007; Knödel et al., 2009; Trapp & Döllner, 2013). The basic idea is to rasterise the target feature and load the raster as texture, and then perform continuous cutting in the image space (Diepstraten et al., 2003). By decoupling the underlying strong topological constraints over the feature geometry, these image-based illustrations avoid much of the computational burden of remeshing and closure. However, despite to what extent of interaction fluency they got, there remain challenges on visual styling during reconstruction. Because these illustrations commonly play with the popular visual schemes – boundary description with separated surface properties – cuttings and closures are mostly performed on boundary surfaces rather than on the individual features. As a result, the way through which the semantic consistency and visual completeness of the objects are retained is unclear, and this is a keen concern for exploratory illustrations. Moreover, the data structures used are very complicated, and the incorporated algorithms are too elaborate, making it difficult to be extended for different illustration styles adaptively. For example, Diepstraten's cut-plane defined that distance texture (Diepstraten et al., 2003) performed cross sections well, but is not suited for cutaways.

For a compact space-partitioned polygonal scene in a virtual building environment, semantics is a fundamental information structure for its key coherency to geometry and topology, and thus it mitigates the complexity of related applications from a top level (Gröger & Plümer, 2012). Illustrative explorations have shown that semantics provides essential guidance for inspecting the target as definition, cutting, closure and styling (Diepstraten et al., 2003; Zhu et al., 2011; Xie et al., 2013; Zhou et al., 2015). CityGML is a broadly accepted international information model, and is well semantically structured. We will mainly work with CityGML's building model, and provide special semantic extensions to its appearance schema.

In this article, we propose a straightforward semantic volume texture (SVT) model, and rely on volume rendering for illustrative exploration of a virtual building. The SVT model is the main contribution of our article. For every discernible building component object with semantic, a unique scalar range is assigned. The spatial pattern designated to each object is embedded through rasterisation (voxelisation) using this scalar, and the rasterised objects together with the pre-extracted semantic hierarchy are then used to synthesise the final SVT. The SVT can exploit the computational efficiency of image-based approaches, while the retained semantic structure over this 'voxel soup' acts as the topology interface, and makes spatial exploration controllable. The semantic tagged spatial pattern is later used as the main visual variable for styling. Because of the true 3D nature of the semantic raster, there is no need for extra closures during reconstruction, which ensures styling consistency as well.

The SVT model is extended to CityGML, and can be seamlessly integrated with CityGML's *appearance model* through the application domain extension (ADE) mechanism. By unleashing most computational and visual styling constraints, more focus can be cast on those cognitive challenges. Later, by working on volume model and volume rendering framework, we demonstrate two illustrative explorations addressing cross-section and cutaway paradigms. The first is a real-time interactive profiling, for which we propose a sectioning lens (S-Lens); the experimental data consist of an LoD4 building model from the CityGML official website. The second is a form of semantic highlighting with surrounding context, where we develop a direct obscured exploration imitating cutaway paradigms. Experimental data for this application are taken from our previous 3D cadastre work (Guo et al., 2013), which involved a scene at a relatively larger scale.

The remainder of this article is structured as follows. Section 2 discusses some related work. Section 3 presents the proposed semantic integrated volume texture extension schema. Implementation details are presented in Section 4, including semantic texture preparation and direct volume rendering (DVR). In Section 5, the proposed SVT model is experimentally validated, and typical application scenarios are demonstrated. Finally, Section 6 integrates our conclusions and some ideas for future work.

## 2. Related work

Direct revelation of special objects or spatial regions from layered building environment has a long tradition, as depicted in Figs. 1 and 2. Some related studies can be traced back to the topic of technical illustrations (Thomas, 1968). Yet computer-generated direct illustration is relatively new (Diepstraten et al., 2003), while direct cross-sectioning or cutting-away of polygonal models suffers from issues such as robustness and high cost (Rossignac et al., 1992; Xie et al., 2013), many researchers have adopted diverse image-based techniques for improved performance (Burns & Finkelstein, 2008; Trapp & Döllner, 2013).

### 2.1. Image-based direct exposing

Diepstraten et al. (2003) initially defined cutting as half-space division and Boolean combinations of half-space divisions. Any boundary object was sampled on graphics hardware according to its signed Euclidean distance to the half-plane (cut-plane); later, this distance image was loaded as texture to perform continuous cutting. Constructive solid geometry (CSG) is an intuitive approach for interactive cutting through concise spatial Boolean conceptions, and image-based CSG (Kirsch & Döllner, 2005; Trapp & Döllner, 2013) Boolean operations and visibility computation (Knödel et al., 2009; Li et al., 2007) are the preferred choices for direct illustration. Yet other researchers use layered depth images (LDIs) (Shade, Gortler, He, & Szeliski, 1998) and its variant volumetric depth sprite (VDS) (Trapp & Döllner, 2008; Trapp & Döllner, 2013) for efficient computation and fast interaction. We share these image-based inspirations and push them as a more formal SVT model. Our volume texture has explicit semantic embedded, and thus differs from Diepstraten's distance image or LDI/VDS's depth

image in that the semantics in SVT act not only as a visual variable, but also as a high-level topological interface. Furthermore, when cutting or extracting a target from a CSG object's image, spatial Booleans are used to remove voxels 'physically', and thus introduce expensive re-arrangements of the whole CSG tree. SVT usually makes the designated voxels fade out virtually and efficiently by ray casting or visibility computations.

### 2.2. Volume graphics and explorative illustration

Volume models and volume rendering have in-depth applications in medical image reconstruction, volume phenomenon emulation and volume illumination simulations (Engel, Hadwiger, Kniss, Rezk-Salama, & Weiskopf, 2006; Jönsson et al., 2013; Rheingans & Ebert, 2001), while volume voxelisation and volume rendering have been widely studied in the field of volume graphics (Kaufman & Mueller, 2005). Mostly, voxelisation was used for the raster-based visibility computations, which could adopt the more efficient image-order algorithm (Kaufman, Cohen, & Yagel, 1993; Kaufman & Mueller, 2005), and the volume texture is the most basic accelerating structure for ray-casting algorithms that directly generate realistic scenes (Kämpe, Sintorn, & Assarsson, 2013; Nielson, 2000). Moreover, the memory consumption of the popular boundary modelling with massive textures and materials today is comparable to that of volume rasters, and the separation of surface and texture requires extra handling during visual manipulation, especially when displacement maps are used (Laine & Karras, 2011). Therefore, there are an increasing number of studies on rasterisation of the whole scene or geometric objects (Eisemann & Décoret, 2006; Forest, Barthe, & Paulin, 2009) with additional lighting effects (Kämpe et al., 2013; Laine & Karras, 2011).

The illustrative exploration of multi-layered and non-invasive volume environments usually adopts magic lenses (Bier, Stone, Pier, Buxton, & Derose, 1993; Ropinski & Hinrichs, 2005) and volumetric lenses (Kruger et al., 2006; Trapp, Glander, & Buchholz, 2008) for direct exposure. Volumetric lenses coincide with the map exploration spirit (Kraak, 1998) of presenting both a detailed focus and the surrounding context (Kruger et al., 2006; Burns & Finkelstein, 2008). In order to illustrate the focus with as little obscurity as possible, the rendering of ROIs or target details is accompanied by various types of highlighting enrichments (Kruger et al., 2006; Glander & Döllner, 2009; Mao et al., 2012; Zhou et al., 2015), as well as data reduction or generalisation of elements outside the ROI. On the basis of the extrusion of planar city blocks, Trapp et al. (2011) developed a multi-level of abstraction (LoA) at city-level generalisation; Zhu et al. (2011) suggested data reduction based on semantics. Many studies emphasise the importance of semantically incorporated visual styling when enhancing and highlighting the focus (Bruckner & Gröller, 2007; Caban & Rheingans, 2008; Rautek, Bruckner, & Groller, 2007). SVT exploits the merits of semantics for both visual styling and spatial operations.

### 2.3. Memory considerations and boundary enhancements

There are two main issues regarding volume texture and volume rendering: excessive memory consumption (Kaufman et al., 1993) and a lack of clear boundaries (Boucheny, Bonneau, Droulez, Thibault, & Ploix, 2009; Trapp & Döllner, 2008). However, recent technical developments may be able to overcome these deficiencies. For example, gradient calculations, environmental lighting and soft shadows can be used to significantly enhance the border effect (Engel et al., 2006); bounding volume hierarchy (BVH), adaptive sparse voxel octree (SVO), out-of-core rendering techniques (Richter & Döllner, 2014) and texture compression can be used to improve memory efficiency (Kämpe et al., 2013; Novák & Dachsbacher, 2012). Recently, a non-uniform texture (NUT) has been reported for modelling and visualisation at an urban scale (Kuang, Chan, Yu, & Wang, 2013). In contrast to their top-to-bottom level division, we are more concerned with semantic integrity in a small-scale environment. Hence, we have adopted rasterisation and texture synthesis in a bottom-to-top manner. As border enhancement and memory efficiency are not the main topics of this article, we do not provide an extensive survey of the literature here, but will describe enhancement in the implementation section, and discuss memory consumption in the validation and conclusion section.

## 3. SVT approach and extension for CityGML

In this section, we first provide the framework description of our proposed SVT-based illustrative visualisation, and then introduce X3D volume texture into the CityGML.

### 3.1. SVT for illustrative visualisation

Image-based CSGs and LDIs are the direct inspirations for SVT, while in the field of X3D graphics, there exist 3D texture standards (Behr & Alexa, 2001; Congote, 2012; X3D, 2013). Although the X3D specification serials lack semantic organisation (Gröger & Plümer, 2012), we can still refer to the introduction of the *appearance model* and bring in its volume texture model. Upon this extended volume texture concept, the technical framework that combines the semantic hierarchy extraction, semantic texturing and volume rendering techniques is shown in Fig. 4.

### 3.2. General considerations of volume texturing

The basic presumption of this article is that every semantic component of the building occupies a specific/non-overlapping 3D physical space. We will use sampling points of this solid, rather than the boundary geometry, to express the semantic distribution. Because of the fact that every component feature usually carries a spatial style at the same time, the distribution of semantics could also be seen as the distribution of spatial patterns, and vice versa. Fig. 5 shows a semantic feature '*wall*' with a repetitive brick-like spatial pattern, where both the feature itself and the spatial pattern can be represented/controlled by the semantic. Spatial patterns could be used to convey volumetric properties such as temperature, humidity, smog, and the general visual style

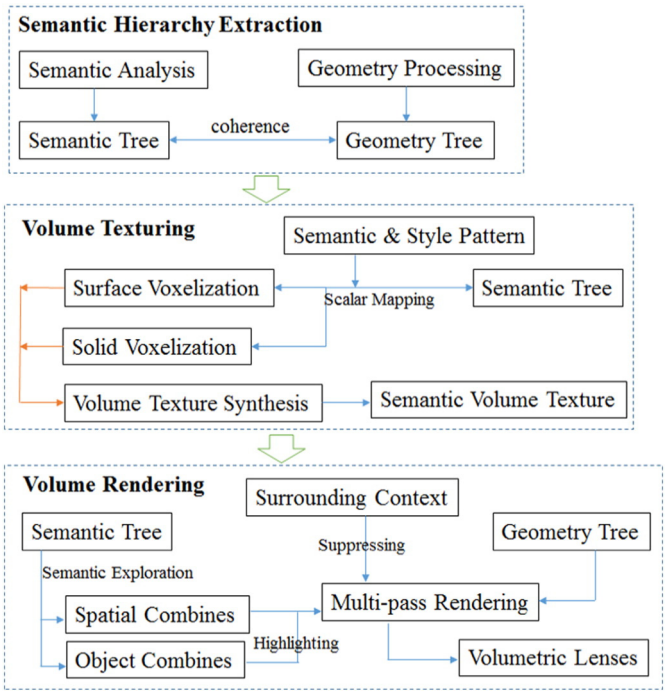


**Fig. 4.** Technical framework of SVT-based illustrative exploration.

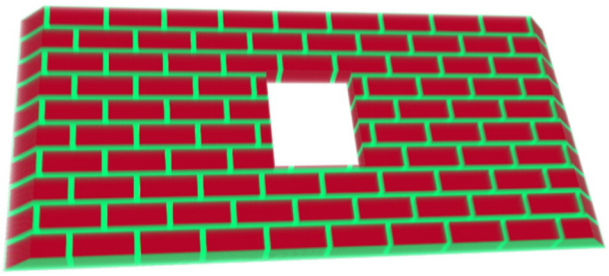

**Fig. 5.** Spatial texture of a building component object wall with repetitive spatial pattern.

distribution confined to the corresponding 3D space. However, for the purpose of explorative visualisation, the volume texture is mainly used to represent the underlying geometry itself in this article, and we go no further emulating the interesting volumetric phenomenon.

Detailed CityGML building model with rich semantics mainly comes from architectural design plans and cadastral registration datasets. We need some sample approaches to convert these semantically embedded boundary geometries into volume texture representation, that is, complete a projection from voxel space ($v = (x, y, z)$) to semantic scalar field ($s$). In order to restore the original spatial structure/pattern in the later rendering stage, the most important concern of sampling is the fidelity. As for regular grid sampling (i.e. volume texture, Kaufman & Mueller, 2005), every cell represents a distribution of the semantic scalar (precisely, the central point of the cell represents this scalar). Presuming that the geometry size of the minimum axial envelope (i.e. axis-aligned bounding box (AABB) space) for the corresponding semantic object is *envelope*, its relationship with the resolution (*dimension*) and grid spacing (*spacing*) is:

$$spacing[i] = envelope[i]/(dimension[i]-1)\ i = 0,\ 1,\ 2.$$

AABB space can avoid the rotation transformations back and forth, thereby sidestepping the introduction of possible numerical round-off error. The larger the sampling density, the smaller the pattern distortion. However, the memory occupied by volume texture increases at an exponential rate. Presuming that the minimum axial distance inside the objects is $d$, the Nyquist–Shannon theorem indicates that the sample spacing should be at least $d/2$ (Hadwiger, Ljung, Salama, & Ropinski, 2008); in practice, the sampling interval is much smaller. However, when the sampling density reaches a certain level, the enhancement of visual saturation is no longer evident (Trapp & Döllner, 2008). On the basis of empirical evidence, when the axial sampling interval is <0.02 m, the volume texture will be suitable for both performance and visual fidelity. For a more sophisticated discussion on rasterisation with error control and sampling theory, please refer to Kaufman and Mueller (2005), Westermann, Sommer, and Ertl (1999) and Hadwiger et al. (2008).

### 3.3. Integrating volume texture with the appearance model

In OpenGL (Open Graphics Library), 3D texture is a series of two-dimensional (2D) images that are equally aligned along the $R$ axis (depth-wise), and the coordinates $S$, $T$ and $R$ correspond to the width, height and depth of the texture, respectively (Fig. 6). The data fields of 3D texture use *1–4* colour components from the red, green, blue, alpha (RGBA) colour space, and the structure of the colour data can also be interpreted as a normalised scalar field. The depth axis runs opposite to the orientation of the volume data, with the positive direction pointing into the screen.

Actual volume data can typically be divided into three categories. The first consists of a series of 2D slices aligned in the axial direction, which in X3D is defined as *ComposedTexture3D*. The second type is integrated image data of the whole volume, such as the *DDS*, *DICOM*

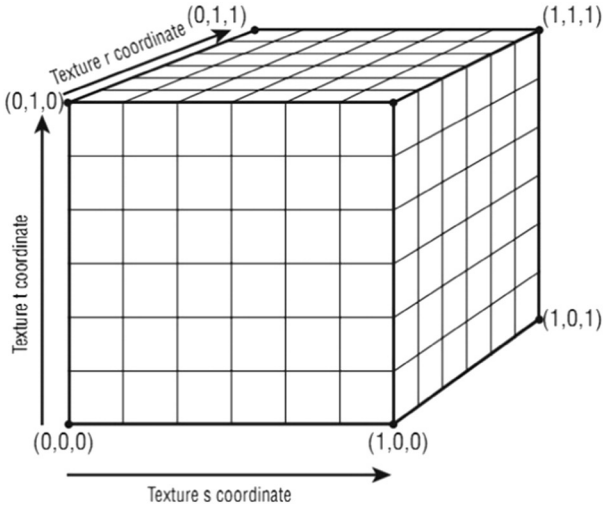


**Fig. 6.** OpenGL 3D texture coordinate system.

and *NRRD* formats. In X3D, these are defined as *ImageTexture3D*. The third category, *PixelTexture3D*, is the simplest raw format of regular grids, and is usually structured as head + serialised data. Because of its structure simplicity, there are a variety of data sources for this texture type, ranging from persistent files to service-based data streams. In order to simplify the discussion, SVT mainly relies on the *PixelTexture3D* definition of data formats. Yet we present an all-around extension schema including all three types of volume data by introducing the X3D volume texture specifications.

In CityGML, a thematic surface attribute attached to a feature is called *appearance data*. These data are mainly *_SurfaceData* composed of X3D materials and textures. The volume texture data are attached to the volume features, and we therefore define the *_VolumeData* structure on the same level as *_SurfaceData*. *_VolumeData* is an abstract class that defines the basic parameters of volume texture, such as scalar type, sampling interval and sampling density. *VolumeTexture* relates to every semantic object, and we use the same syntax '*target*' of the appearance model to specify the identification (*gml*::*id*) carried by the geometric object. The most important field of the *VolumeTexture* class is the *scalarRange* used by the semantic object. This scalar range will be directly used to control the visualisation exploration. The unified modelling language (UML) diagram of volume texture extension for the appearance model is shown in Fig. 7, where extended new classes are enclosed by a red-dashed polygon.

## 4. Semantic volume texturing and volume rendering

In this section, we describe the implementation details listed in Fig. 4 by keeping to the topics around semantic volume texturing and volume rendering. These two aspects are the fundamental basis of direct illustrative exploration.

### 4.1. Semantic and geometry hierarchy extraction

The CityGML data model is organised according to semantics, which are built on the geometric primitives of GML. The semantic primitives of the virtual building model are primarily space features with volume; the topological relationship between these semantic primitives formed through aggregation can be divided into three types: aggregate, complex and composite (ISO-19107/ISO-19109). Structural components of aggregate objects share no topological relationships. The complex type is defined by a cell complex; its structural components must be disjoint, must not overlap and are allowed to touch, at most, at their boundaries or to share parts of their boundaries (OGC, 2012). The composite type is just the complex type confined to objects with the same dimension.

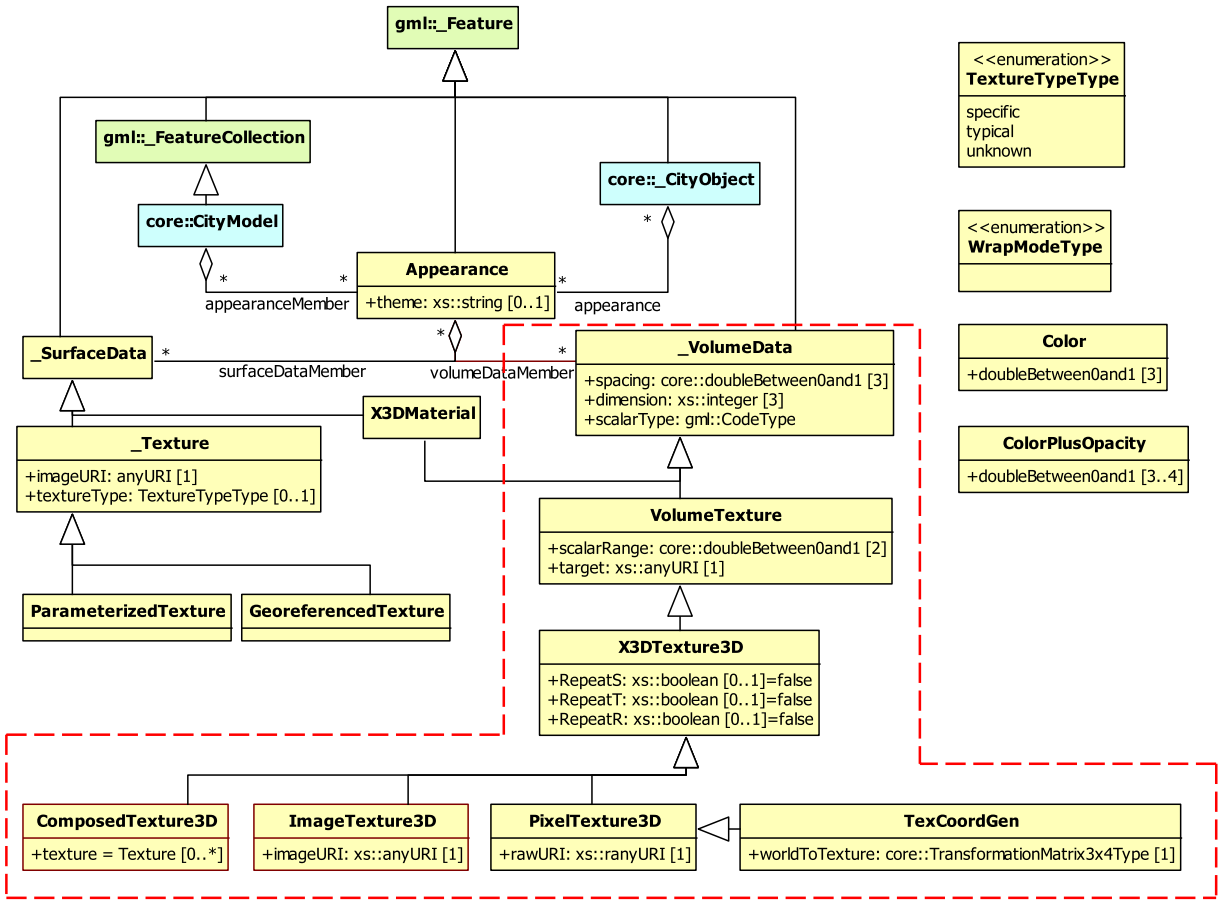


**Fig. 7.** UML diagram of volume texture extension to appearance model in CityGML. Classes enclosed by a red-dashed polygon are the extended new ones. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Complex and composite are the main topological forms of basic building features, while the combination of semantics can be divided into two types: hierarchical relationships and aggregation relationships (Zhu et al., 2011). These objects and their relationships might follow the Extensible Markup Language (XML) schema definition, and are tracked out from the *gml* file(s) to establish the semantic tree. Regardless of the actual data quality processing, the main work required to construct the semantic hierarchy is as described below:

> For an aggregate or complex object, if it passes the closing test, we complement the *XLink*-shared faces and record it as a solid structure. However, if it does not form a closed entity, but has solid semantics (e.g. a barn), we complement the open boundaries and record it as a *ClosureSurface* entity. If the object does not form a solid, but has actual semantics, it is recorded as a surface structure; for every semantic and geometric object, we establish the coherent relationship via *gml*::*id*. Because the geometric layer of GML is mainly defined by cell complexes, the object's surface is usually a polygonal structure, and may require further triangulation or (constrained) Delaunay triangulation.
>
> We store the extracted triangular meshes in a semantic tree.

### 4.2. Semantic voxelisation

For an independent city building, the semantics of the aggregating components constitute the multi-valued scalar distribution of the space. We use binary voxelisation to complete the projection from semantic field ($s$) to voxel space ($v$). In order to ensure the semantic integrity of these components, we propose a bottom-to-top method of semantic synthesis.

#### 4.2.1. Surface voxelisation and solid voxelisation

The semantic object of a city building is mainly a solid object with a specific volume, whereas there are also many open geometric objects. In particular, the boundaries or silhouettes of target features are effective characteristics describing the surrounding context, and these are usually recorded as surface structures. Therefore, we must provide both surface voxelisation and volume voxelisation.

In general, binary voxelisation can be described via a point location algorithm as follows:

i. Divide the AABB space of the target into a series of regular cells according to sampling density and spacing; prepare an appropriate VolumeBuffer array;
ii. Split the target object into a series of tetrahedrons;
iii. Repeat the following:
   (a) For every cell, take the geometric centre as the sampling point, and locate the simplex tetra that the sampling point is inside/on;
   (b) According to the value returned from (a), fill the array of scalar values for the cell;
iv. Write the VolumeBuffer to file.

In step iii(a), various multi-level octree techniques can be adopted to accelerate the point location, and in this case, the target object is not necessarily divided into simplices. For a rigorous discussion, the point

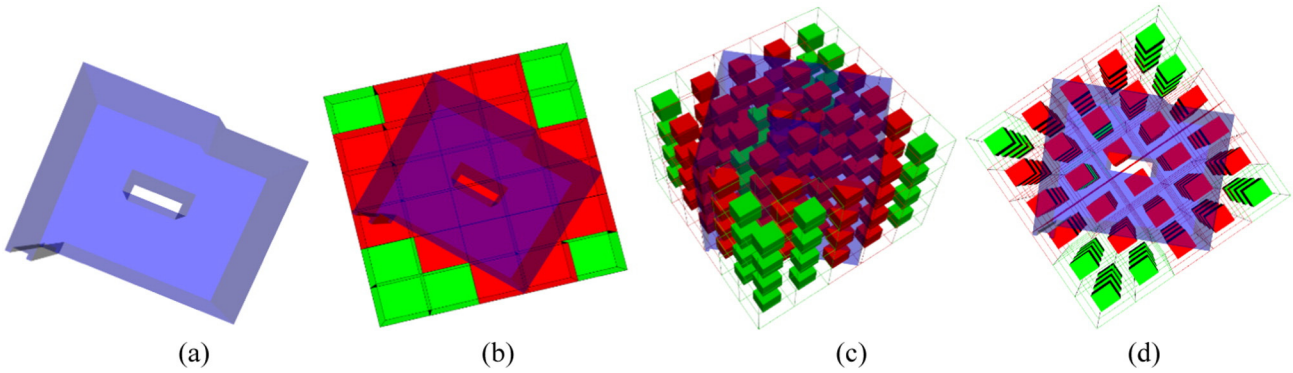


**Fig. 8.** Surface voxelisation and solid voxelisation. (a) An elevator well model (of a single storey), (b) the AABB space of the model divided into uniform cells; cells that intersect with the model boundary are rendered in red, and cells that do not intersect are rendered in green. (c) Shrunken cells are used to expose the inner situation, (d) top view of (c). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

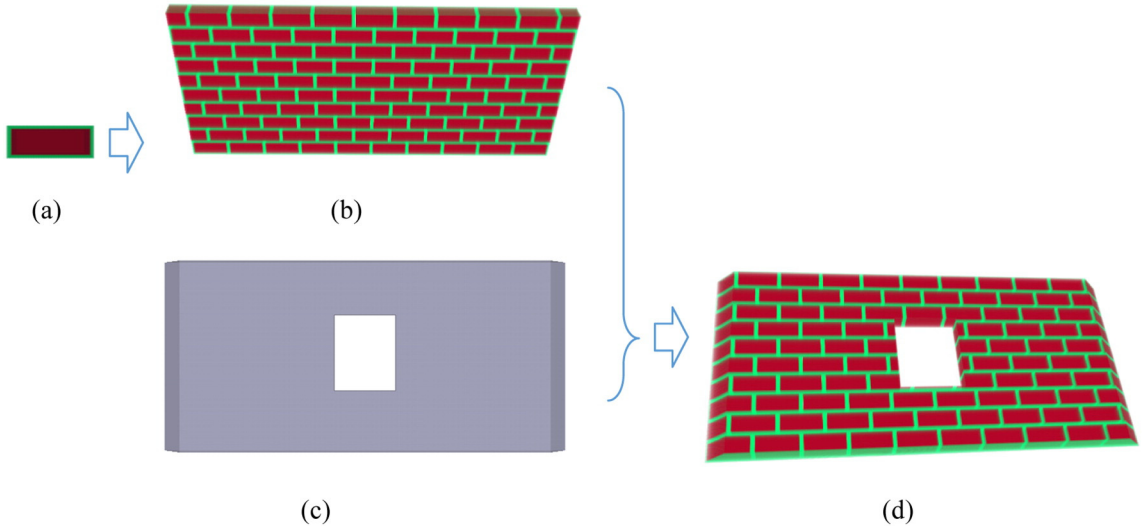


**Fig. 9.** Semantic and spatial pattern texturing. (a) Base spatial pattern. (b) Pattern repeated along height and depth directions to fill the AABB space of the semantic object. (c) Comparison geometry stencil. (d) By referring to the comparison stencil, the two-step voxelisation produces the semantic embedded volume texture of a wall object.

location results in iii(a) can be defined by Boolean tests between a single-cell box and the specific boundary geometry of the target object. If they intersect and the intersection is the cell itself, then the sampling point is located inside the cell; if they intersect but the intersection is not the cell itself, the sampling point is located on the boundary. Therefore, solid voxelisation and surface voxelisation can be achieved by the same routine.

However, strict Boolean computations over geometries analytically have a high cost in general. Therefore, we can use collision-detect between the boundaries and the cell boxes instead. Efficient collision detection could employ the *separating axis theorem* to complete the overlapping calculation. A surface voxelisation using fast collision-detect is illustrated in Fig. 8, with coarse sampling density at a resolution of $6 \times 5 \times 7$.

#### 4.2.2. SVT synthesis

When voxelising objects in the semantic tree, we must consider mapping the spatial pattern of the object. One direct spatial pattern mapping can be considered as a two-step binary voxelisation:

i. Fill the AABB space of the semantic object with a styled pattern;
ii. Voxelise the semantic objects, with cell scalars modified to styled texture or background noise according to the collision results.

This gradual process takes the object's boundary geometry as a comparison stencil. If the collision test indicates that the sampling cell is inside the object, the scalar value of the cell takes the scalar value of the styled texture; otherwise, it takes the background noise value. Fig. 9 demonstrates the mapping process of a repetitive styled texture. Note that this repetitive texture cannot simply be completed through the OpenGL '*GL_REPEAT*' command.

The semantic texture of every target feature occupies a basic locally positioned AABB space (although CityGML objects are generally absolute geo-positioned), and these AABB spaces will cross and overlap. In order to synthesise the final SVT, we first place each AABB space to its relative position in the building texture space. Next, we apply raster Boolean operations to complete the texture synthesis. These raster Booleans are usually bit operations over the boundary geometry stencil again. Fig. 10(a) shows the first two steps of the semantic synthesis in the LoD4 model,[1] and Fig. 10(b) shows the final SVT of this model.

The geometry stencil comparison strategy and the bottom–up synthesis of SVT warrant the complete and consistent embedding of semantics and spatial patterns, but this would entail appreciable computational costs. For the innermost voxelisation process (Section 4.2.1), it would take $O(n * m)$ time to complete a pairwise comparison between every

[1] http://www.citygml.org/fileadmin/count.php?f=fileadmin%2Fcitygml%2Fdocs%2FAC11-FZK-Haus_LOD4_SpaceSolid_v1.0.0.zip.

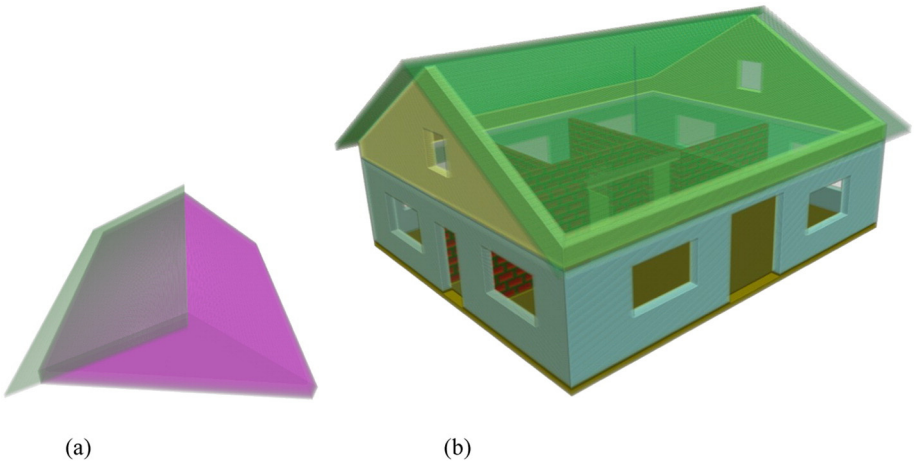


**Fig. 10.** SVT synthesis. (a) The first two steps of raster Boolean composition, a roof and an adjacent room. (b) The final semantic volume texture of an LoD4 building model; here rooms are rendered totally transparent.

grid cell (voxel) and tetrahedron, where $n$ is the number of grid cells and $m$ is the number of tetrahedrons. The fast but imprecise 3D scan conversion (Dong, Chen, Bao, Zhang, & Peng, 2004; Wang & Kaufman, 1993) improves the voxelisation to nearly $O(n)$, that is, the upper bound of the rasterisation complexity. Using this $O(n)$ index, however, the two-step spatial pattern embedding would consume $O(n^2)$ time, and the final SVT synthesis would consume a further $O(n^3)$ time, for they both use the exact pairwise stencil comparison approach. Therefore, it is necessary to design variable grained parallel voxelisation on graphics processing unit (GPU) hardware exploiting merits of CUDA or OpenCL architecture (Schwarz & Seidel, 2010).

Thus far, we have only discussed objects that can be voxelised. In CityGML, there are objects that have fine scales of far <0.2 m in terms of LoD4 point positioning accuracy. These are mainly *Furniture* and *Installation* objects. For these two classes, we can apply data reduction methods or a vector-integrated volume rendering technique.

### 4.3. Direct volume rendering

The scalar value mapped to each voxel represents semantic as well as spatial patterns, and forms the multi-valued/multi-layered 3D scalar field. In order to render these volumetric data, DVR techniques use ray casting to emulate lighting effects by calculating every voxel's contribution for the final image synthesis. Thus, volume rendering prevails surface rendering, because in the latter situation, light interaction is only calculated on the boundary (Hadwiger et al., 2008). A typical GPU-accelerated ray-casting algorithm is as follows:

i. Load 3D textures representing the volumetric data, as well as a one-dimensional (1D) texture representing the transfer function, into texture memory. The 1D texture is an index array that maps scalar to colour and opacity.
ii. Create the proxy bounding box of the volume. The back faces of this proxy geometry are used to compute the termination point of the ray casting.

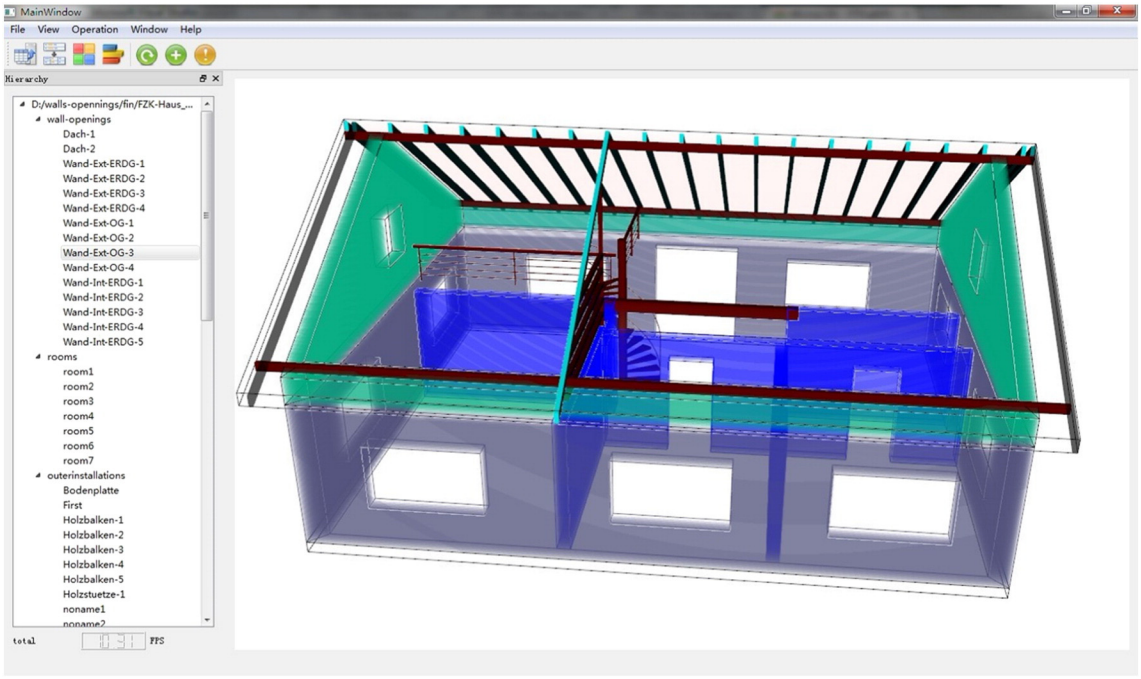


**Fig. 11.** The main user interface showing wireframe-integrated volume rendering of an LoD4 building model.

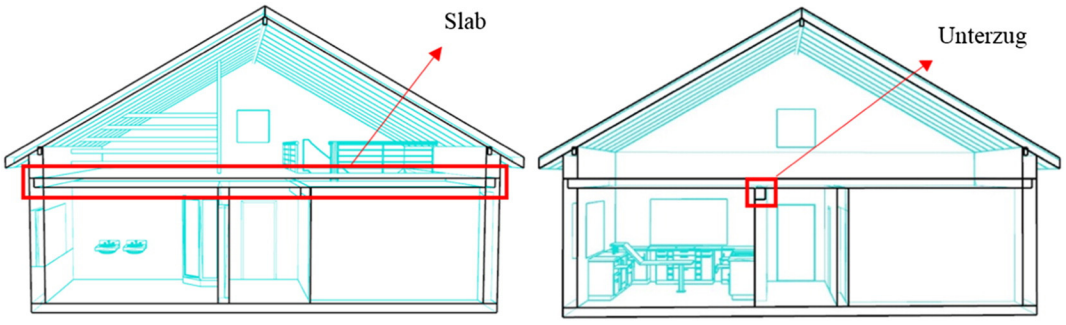


**Fig. 12.** Cross section of the LoD4 surface model in SketchUp, sectioning lines are highlighted in bold dark. Left, front view; right, back view. The hidden objects emphasised by red rectangles (with tagged names *Slab* and *Unterzug*) are generally difficult to distinguish from surrounding structural lines. This is the classic problem of visual clutter. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

iii. For each pixel on the proxy geometry's front faces, a fragment shader evaluates the ray casting through the 3D texture. The scalar value sampled from the 3D texture is used as the index to the colour and opacity combination in the 1D texture. Ray traversal is terminated when either it reaches the back face or full opacity is achieved.

The semantic scalar is interpreted as the main visual variable through the transfer function. DVR develops plenty of methods to control the behaviour of voxels, with semantic or spatial operations on target objects accomplished by changing the transfer function. High-level semantic and spatial operation for flexible illustration of SVT will be demonstrated in Section 5.

As for border enhancement, there are various local and global illumination techniques (Jönsson et al., 2013). These additional lighting effects commonly rely on gradient or curvature calculations. We could also employ geometry-integrated rendering, which is of merit when accurate geometry boundaries are essential in building model illustrations. Fig. 11 illustrates this technique on the aforementioned LoD4 SVT model. Note that the indoor rooms are rendered transparent to simulate the usual *appearance* scheme, while the geometry wireframe (but not geometry surface) is integrated to show the accurate border.

## 5. Illustrative exploration applications

After semantic voxelisation of the virtual building model, we have a complete description of the spatial pattern for the targets. This enables a consistent combination both spatially and semantically, which is constantly needed in illustrative explorations. In this section, by working on volume rendering techniques, we demonstrate typical direct illustrative applications through semantic and spatial operations on SVT voxels.

### 5.1. Real-time cross-sectioning

The cross-section diagrams that exploded view and slide-out drawer strategy (Ropinski & Hinrichs, 2005; Li, Agrawala, Curless, & Salesin, 2008; Vandysheva et al., 2012) are similar exploration approaches that directly remove visual obstructions caused by the third dimension. In particular, real-time continuous cross-sectioning is often desirable.

The sectioning plane can generally be expressed as:

$$Ax + By + Cz + D = 0.$$

As for the SVT, semantic voxels lie above the cutting plane, so that:

$$Ax + By + Cz + D > 0.$$

Therefore, in the DVR shader, we can start the cut operation by substituting the voxel coordinates into this inequality. By simply setting them to total transparency, we virtually 'remove' the corresponding portion above the cutting plane. Without any additional cost on subsequent operations as topological reconstruction, *clip-surface* closures, and styling, this cross-sectioning achieves an efficiency that is unattainable by geometric cutting and reconstructing operations.

The experimental data in this section are from an LoD4 indoor model with distinctive Building Information Modelling (BIM) features. We load this model into Trimble's SketchUp (formerly Google SketchUp), and perform a cutting operation. As illustrated in Fig. 12, after cutting, open boundaries are formed at the *clip-surface*, where thick lines have been added to identify the cut. By contrast, on the cross section of the semantic raster, as illustrated in Fig. 13(a), the solid object remains solid. Therefore, hidden objects with the tagged names *Slab* and *Unterzug* can clearly be depicted along with nearby components. The perception of the spatial relationship among *Slab*, *Unterzug* and walls

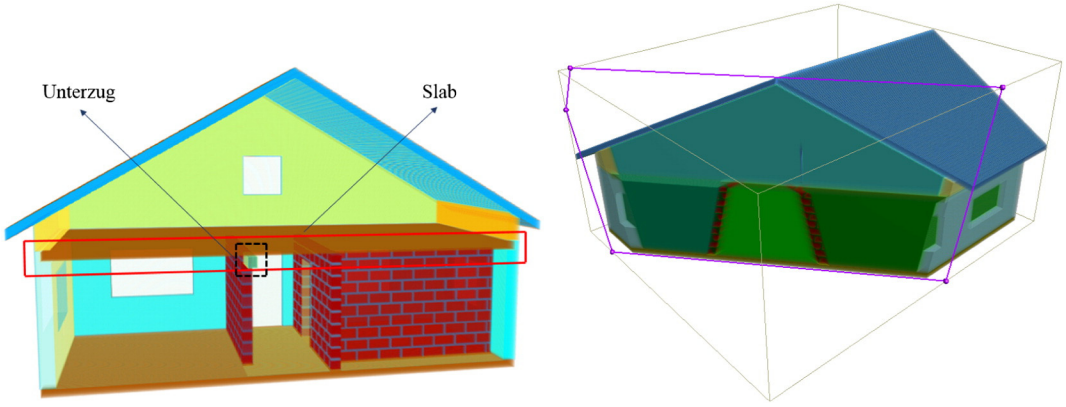


**Fig. 13.** Cross-sectioning on SVT model. (a) Basic cross-sectioning, (b) interactive continuous sectioning by arbitrary plane. Here, the hidden objects with tagged names *Slab* and *Unterzug* are clearly styled, and can thus be positively distinguished from the surrounding objects.

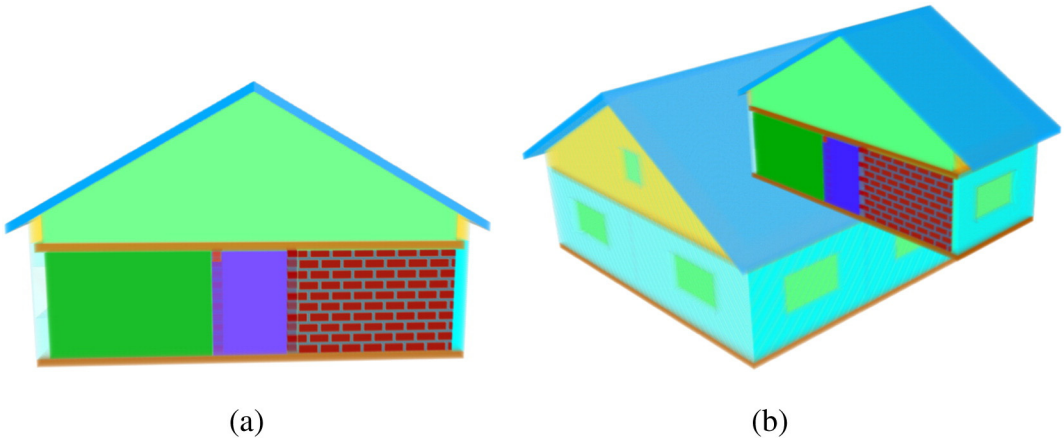


**Fig. 14.** Sectioning lens (S-Lens), cross-section diagram with contextual information. (a) Cross-sectional view, (b) image synthesis by multi-pass volume rendering, with contextual information added to provide positioning cues.

is not obscured by the front layers. Fig. 13(b) shows a screenshot in which an arbitrary plane has been applied to dissect the building. This dynamic continuous cutting releases valuable motion cues (Kraak, 1993) to better inspect the hidden spatial relationships.

Single cross-section diagrams often lose global information about spatial relationships, that is, contextual information for positioning. Using the idea of volumetric lenses (Ropinski & Hinrichs, 2005), we propose a profiling/sectioning lens (S-Lens). This lens first considers the clipping operation on the cutting plane, and then sequentially renders the overall volume texture and the intercepted portion. In the final image, the pixels of the intercepted body cover the pixel formed by the previous global voxels. This produces an image with both local profile details and global surrounding context. Fig. 14 illustrates this process and effect.

### 5.2. Semantic highlighting on-the-fly

3D cadastre focuses on space usage and registration from different legal aspects (van Oosterom, Stoter, & Lemmen, 2005). 3D parcels of ownership are of utmost concern, and constitute the most important content of virtual building environments. Actual architectural design plans provide plenty of detail for space partitions to implement a 3D ownership subdivision. We extract an intermediate data model from BIM Industrial Foundation Class (BIM-IFC) specifications. This model continues our previous work on 3D cadastres (Guo et al., 2013), and it retains the major geometric features of the architecture, while the private indoor details are substantially simplified. Moreover, according to the CityGML specifications, we build a compatible semantic description for this data model. An instance of this model is shown in Fig. 15. The actual building is located in Shenzhen, People's Republic of China. Fig. 15(a) shows the model provided by surface rendering, and Fig. 15(b) shows the volume rendering effect of SVT. It could be seen that the aggregation of 3D parcels forms a typical scenario of visual occlusion.

In terms of applications and cadastral aspects, semantic information provides a natural language interface for planning, managing and inspection. As for visual exploration, semantic combinations from user interaction are of basic functionality. For the data model in Fig. 15, after semantic voxelisation, we obtain an independent mapping from the semantic scalar to visual variables, allowing flexible but consistent visual control via semantics. Fig. 16 shows the dynamic combination of the semantic objects of interest, where two incident apartments with adjacent infrastructures are highlighted.

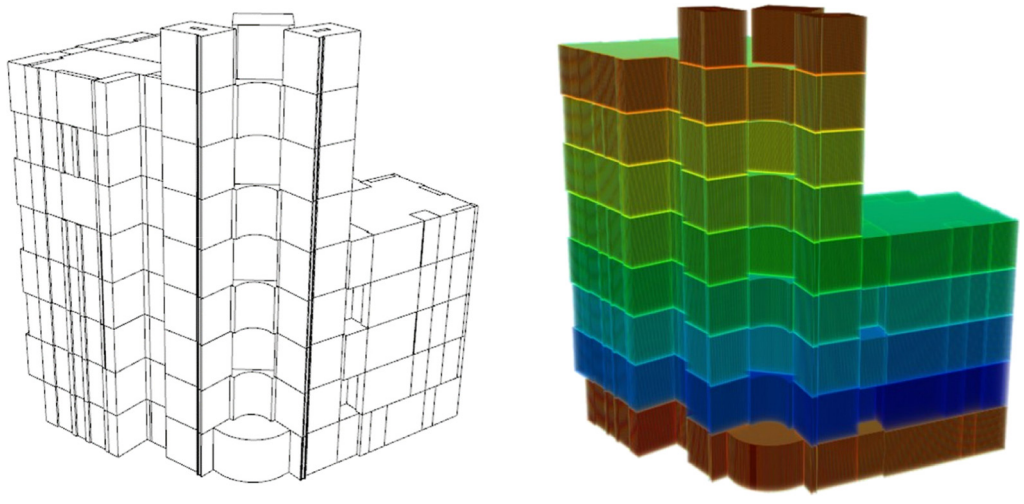


**Fig. 15.** Ownership subdivision on a 3D cadastral model in Shenzhen, People's Republic of China. Either from surface rendering (a) or volume rendering (b) perspective, the visual occlusion and topological coupling layers prevent implementation of an easy inspection scheme. (a) Property space provided by mesh model. (b) Property space provided by SVT model.

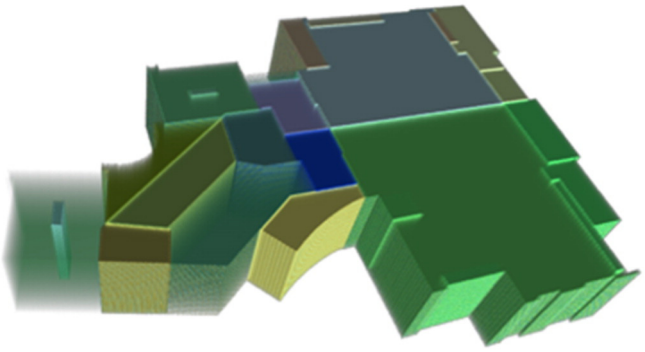

**Fig. 16.** Dynamic combination of semantic targets showing the spatial layout of two incident apartments. These apartments are singled out and highlighted from the semantic interface. This highlighted combination is efficient and consistent because of the independent spatial pattern mapping.

Solving the visual style mapping for the target object with dynamic combinations is the first step. In order to properly illustrate these targets of interest from visual obstruction, context generalisation and synthesis must be used (Glander & Döllner, 2009; Zhu et al., 2011). This is a far more complicated process, involving cognitive and cartographic communication. Cutaway paradigms dissolve this process into a couple of relatively simple treatments, as introduced in Section 1. However, the cutaway views observed from the exquisite peeling window are too factitious. Here, we propose a new approach that imitates cutaway illustrations. Instead by viewing through a cut window, this semantic highlighting presents a direct exposure diagram with global context. Similar to Kruger et al.'s (2006) approach, it comprises three stages of processing:

i. Through the step-by-step shell extraction of each semantic object, we synthesise the hierarchical shell of the whole building to be used as the surrounding context ($v_{sc}$).

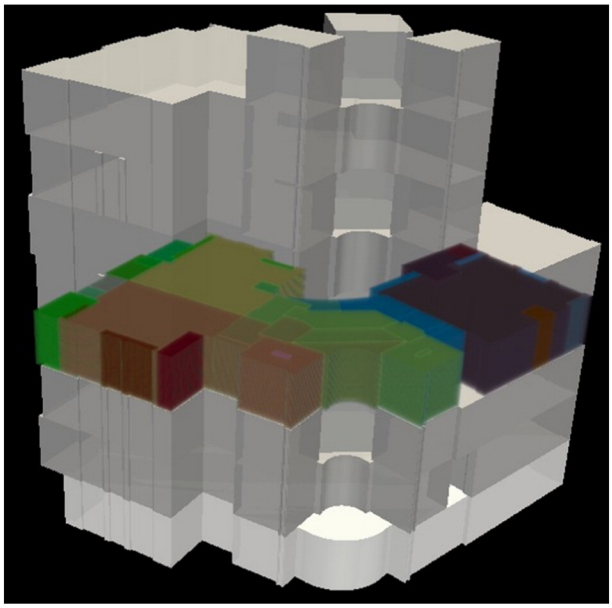

**Fig. 18.** Clear view of highlighted semantic objects with contextual surroundings. This effect composes treatments such as semantic highlighting, surrounding context extraction and a final image synthesis through multi-pass volume rendering.

ii. The semantic target $v_{obj}$ is extracted (as shown in Fig. 16) and a visual style prepared.
iii. Render $v_{sc}$ and $v_{obj}$ in sequence to synthesise the final image.

Fig. 17 illustrates the process of shell extraction. Note that this is also a data reduction process, and the shell object $v_{sc}$ can usually be prepared in advance. Fig. 18 shows the final effect of the total treatment. Because of the mechanism of light casting, visual blocking by outside layers is

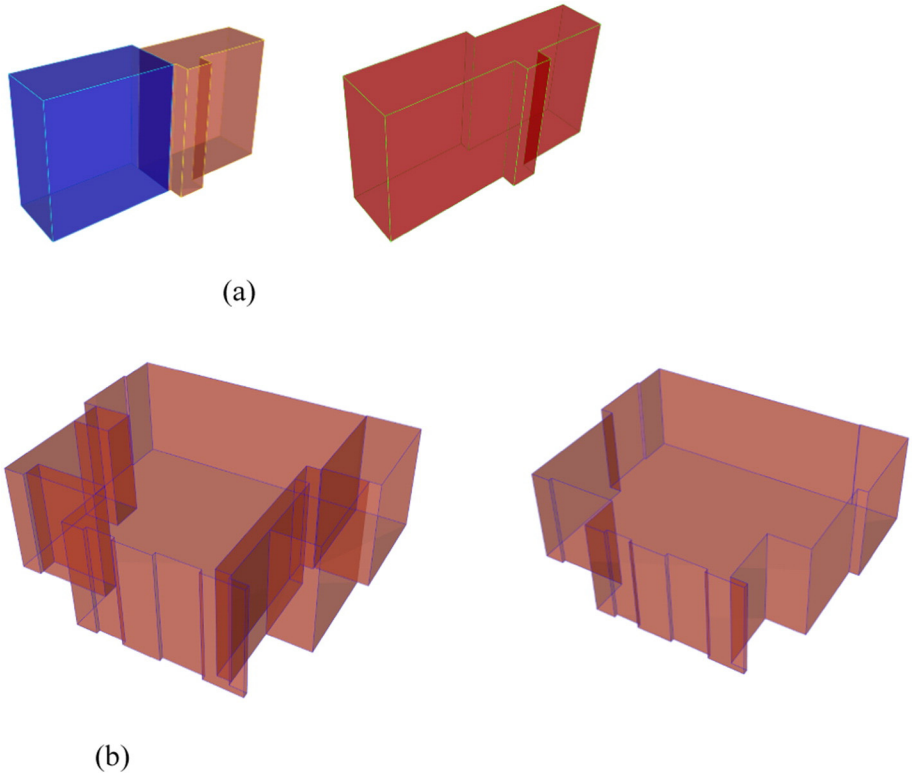


**Fig. 17.** Hierarchical shell extraction. (a) Shell extraction by merging fine-grained components. (b) Shell extraction of an apartment object.

**Table 1**
Memory and rendering performance.

| Model | Sample density | Sample spacing (m) | Memory consumption (MB) | Frames per second (fps) | |
|---|---|---|---|---|---|
| | | | | DVR | Wireframe integrated DVR |
| LoD4 building | 271 × 230 × 163 | 0.048 × 0.048 × 0.040 | 9.69 | 60+ | 60+ |
| | 800 × 700 × 400[1*] | 0.016 × 0.016 × 0.016 | 214.46 | 42.26 | 10.31 |
| 3D cadastre parcel | 300 × 300 × 300 | 0.076 × 0.13 × 0.12 | 25.75 | 60+ | 55.84 |
| | 800 × 800 × 300[2*] | 0.028 × 0.047 × 0.12 | 183.11 | 47.48 | 17.96 |
| | 1000 × 1200 × 600 | 0.023 × 0.032 × 0.06 | 686.65 | 1.13 | – |

Illustrations in real-time cross section used the 1* configuration, while Semantic Highlighting illustrations used 2* configuration. '–' represents we did not take that test.

eliminated. In addition, because of the two dependent multi-pass volume renderings, we can use different density textures for the foreground and background. That is, for the highlighted target object, fine-grained SVT is used; for suppressed background, normal-resolution SVT is used. This kind of non-uniform texture can alleviate the problem of memory consumption to a certain degree.

### 5.3. Memory and rendering efficiency

For the aforementioned two data models, we tested the illustrations on different sampling density SVTs using an NVIDIA GTX760 GPU with 2 GB video memory and an Intel I3 2120 CPU with 4 GB main memory, the running OS was a X64 Windows 7. The results are listed in Table 1.

Memory consumption varies significantly with the sampling rate. As for the LoD4 model, a 271 × 230 × 163 sample density consumes 9.69 MB, whereas an 800 × 700 × 400 sampling density consumes over 200 MB abruptly. The 60+ frames per second (fps) rendering performance indicates an excellent interaction fluency in multi-threading environment. However, on our PC-based testing platform, when the texture memory approached 600 MB, performance with interactive illustrations degraded drastically. This was mainly caused by the data-swapping bottleneck between storage and memory. Moreover, the wireframe geometry-integrated DVR reduced the rendering performance badly, as it consumes additional resources for the visibility computations.

## 6. Conclusions

Burns and Finkelstein (2008) summarised the main challenges of 3D visualisation as visual occlusion, visual clutter and visual navigation/exploration. Illustrative visualisations had shown their value by directly removing the visual occlusion and providing navigational information through preserving positioning contexts. Although image-based direct illustrations pushed exploration forward by commutating computational burdens through voxelisation, they usually lack a clear description for keeping semantic and visual consistent. By addressing the inherent deficiency of the boundary geometry with separated surface description, we proposed an SVT model. This true-3D raster model integrated spatial patterns as well, thus warranting semantic consistency and complete visual styling during cutting and reconstructing operations in direct illustrations. All these features provide a good basis for a better resolution of the cognitive challenges confronted by visual exploration in densely clustered virtual building environments. Later, we described the necessary approaches for the preparation of the SVT, and used volume rendering techniques to demonstrate applications of cross-section and cutaway illustrations. The experimental results show that our SVT model is suitable for efficient spatial explorations.

When working with volume model and volume rendering, the main concern has always been the contradiction between memory occupation and rendering speed. Numerous studies have examined BVH, SVO, NUT, and texture compression to improve the memory efficiency; however, the number of voxels that can be processed by GPUs always has an upper limit over an interval, and volume memory exceeding this limit would significantly affect the total rendering performance. Meanwhile, with respect to micro-scale dwelling spaces, as Shojaei et al. (2013) have commented, it is more common to inspect an interested property parcel locally, and there is no need to render all visible city objects at one time. And, spatial details with refined geometry resolution (in CityGML case, of far <0.02 m) are not especially useful for conveying the feature shape. Taking into account these observations, we mainly tested plans that combine a fine target texture and coarse background texture to reduce memory consumption. Experiments showed that the overall memory consumption of SVT is acceptable, and direct illustrations are applicable on common hardware.

However, BVH/SVO, out-of-core renderings and NUT techniques have considerable experience in handling large-volume data, such as that of oil and gas, meteorology and large-scale finite element models (FEMs). This experience may allow us to promote volume texture and volume rendering to large scenes. SVT-based applications and evaluations incorporating BVH/SVO/NUT techniques at city scale are one direction of interest, as we believe direct illustrations would be useful for revealing spatial phenomena at a larger scale. In addition, volumetric phenomena such as temperature, humidity, fire, smog and air circulation in micro-scale indoor environments affect the quality of life of almost everybody. Thus, illustrations and simulations of these spatial phenomena would be another direction for future consideration.

## Ackowledgement

The research for this paper was funded by the National Natural Science Foundation of China (No. 41471325) and the National Key Technology R&D Program of the Ministry of Science and Technology of China (No. 2012BAB16B01). The authors appreciate all the three anonymous reviewers for their valuable and constructive comments. Special thanks are also addressed to Associate Editor Yu Liu who managed the submission.

## References

Behr, J., & Alexa, M. (2001). Volume visualization in VRML. *Proceedings of the Sixth International Conference on 3D Web Technology* (pp. 23–27). ACM.
Bier, E.A., Stone, M.C., Pier, K.A., Buxton, W., & Derose, T.D. (1993). Toolglass and magic lenses: The see-through interface. *Annual Conference on Computer Graphics* (pp. 73–80).
Boucheny, C., Bonneau, G. -P., Droulez, J., Thibault, G., & Ploix, S. (2009). A perceptive evaluation of volume rendering techniques. *ACM Transactions on Applied Perception, 5*, 1–24.
Bruckner, S., & Gröller, M.E. (2007). *Style transfer functions for illustrative volume rendering. Computer Graphics Forum.* (pp. 715–724)Wiley Online Library, 715–724.
Burns, M., & Finkelstein, A. (2008). Adaptive cutaways for comprehensible rendering of polygonal scenes. *ACM Trans. Graph., 27(5).* (pp. 154). ACM.
Caban, J.J., & Rheingans, P. (2008). Texture-based transfer functions for direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics, 14*, 1364–1371.
Congote, J. (2012). Medx3dom: Medx3d for x3dom. *Proceedings of the 17th International Conference on 3D Web Technology* (pp. 179-179). ACM.
Diepstraten, J., Weiskopf, D., & Ertl, T. (2003). Interactive cutaway illustrations. *Computer Graphics Forum, 22*, 523–532.
Dong, Z., Chen, W., Bao, H., Zhang, H., & Peng, Q. (2004). Real-time voxelization for complex polygonal models. *Proceedings. 12th Pacific Conference on Computer Graphics and Applications, 2004. PG 2004* (pp. 43–50). IEEE.
Eisemann, E., & Décoret, X. (2006). Fast scene voxelization and applications. *ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (pp. 71–78).

Engel, K., Hadwiger, M., Kniss, J.M., Rezk-Salama, C., & Weiskopf, D. (2006). *Real-time volume graphics*. Natick: Ak Peters.
Forest, V., Barthe, L., & Paulin, M. (2009). Real-time hierarchical binary-scene voxelization. *Journal of Graphics, GPU, and Game Tools*, *14*, 21–34.
Glander, T., & Döllner, J. (2009). Abstract representations for interactive visualization of virtual 3D city models. *Computers, Environment and Urban Systems*, *33*, 375–387.
Goetz, M. (2012). Towards generating highly detailed 3D CityGML models from OpenStreetMap. *International Journal of Geographical Information Science*, *27*, 845–865.
Gröger, G., & Plümer, L. (2012). CityGML — interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, *71*, 12–33.
Gross, M., & Pfister, H. (2007). *Point-based graphics*. Morgan Kaufmann.
Guo, R., Li, L., Ying, S., Luo, P., He, B., & Jiang, R. (2013). Developing a 3D cadastre for the administration of urban land use: A case study of Shenzhen, China. *Computers, Environment and Urban Systems*, *40*, 46–55.
Hadwiger, M., Ljung, P., Salama, C.R., & Ropinski, T. (2008). Advanced illumination techniques for GPU volume raycasting. *ACM SIGGRAPH ASIA 2008 courses* (pp. 1). ACM.
Hong, S., Jung, J., Kim, S., Cho, H., Lee, J., & Heo, J. (2015). Semi-automated approach to indoor mapping for 3D as-built building information modeling. *Computers, Environment and Urban Systems*, *51*, 34–46.
Isikdag, U., Zlatanova, S., & Underwood, J. (2013). A BIM-oriented model for supporting indoor navigation requirements. *Computers, Environment and Urban Systems*, *41*, 112–123.
Jönsson, D., Sundén, E., Ynnerman, A., & Ropinski, T. (2013). A survey of volumetric illumination techniques for interactive volume rendering. *Computer Graphics Forum*, *33*, 27–51.
Kämpe, V., Sintorn, E., & Assarsson, U. (2013). High resolution sparse voxel dags. *ACM Transactions on Graphics*, *32*, 101.
Kaufman, A., Cohen, D., & Yagel, R. (1993). Volume graphics. *Computer*, *26*, 51–64.
Kaufman, A., & Mueller, K. (2005). *Overview of volume rendering*. The visualization handbook, 127–174.
Kirsch, F., & Döllner, J. (2005). OpenCSG: A library for image-based CSG rendering. *Proceedings of the Annual Conference on USENIX. Annual Technical Conference* (pp. 129–140). Anaheim, CA: USENIX Association.
Knödel, S., Hachet, M., & Guitton, P. (2009). Interactive generation and modification of cutaway illustrations for polygonal models. In A. Butz, B. Fisher, M. Christie, A. Krüger, P. Olivier, & R. Therón (Eds.), *Smart graphics* (pp. 140–151). Berlin Heidelberg: Springer.
Kraak, M. -J. (1993). Three-dimensional map design. *The Cartographic Journal*, *30*, 188–194.
Kraak, M. -J. (1998). The cartographic visualization process: From presentation to exploration. *The Cartographic Journal*, *35*, 11–15.
Kruger, J., Schneider, J., & Westermann, R. (2006). Clearview: An interactive context preserving hotspot visualization technique. *IEEE Transactions on Visualization and Computer Graphics*, *12*, 941–948.
Kuang, Z., Chan, B., Yu, Y., & Wang, W. (2013). A compact random-access representation for urban modeling and rendering. *ACM Transactions on Graphics*, *32*, 1–12.
Laine, S., & Karras, T. (2011). Efficient sparse voxel octrees. *IEEE Transactions on Visualization and Computer Graphics*, *17*, 1048–1059.
Li, W., Agrawala, M., Curless, B., & Salesin, D. (2008). Automated generation of interactive 3D exploded view diagrams. *ACM Transactions on Graphics*, *27*, 1–7.
Li, W., Ritter, L., Agrawala, M., Curless, B., & Salesin, D. (2007). Interactive cutaway illustrations of complex 3D models. *ACM SIGGRAPH 2007 papers* (pp. 31). San Diego, California: ACM.
Mao, B., Harrie, L., & Ban, Y. (2012). Detection and typification of linear structures for dynamic visualization of 3D city models. *Computers, Environment and Urban Systems*, *36*, 233–244.
Nielson, G. (2000). Volume modelling. In M. Chen, A. Kaufman, & R. Yagel (Eds.), *Volume graphics* (pp. 29–48). London: Springer.
Novák, J., & Dachsbacher, C. (2012). Rasterized bounding volume hierarchies. *Computer Graphics Forum*, *31*, 403–412.
OGC (2012). *OGC city geography markup language (CityGML) encoding standard 2.0*. Open Geospatial Consortium, Inc.
Rautek, P., Bruckner, S., & Groller, E. (2007). Semantic layers for illustrative volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, *13*, 1336–1343.
Rheingans, P., & Ebert, D. (2001). Volume illustration: Nonphotorealistic rendering of volume models. *IEEE Transactions on Visualization and Computer Graphics*, *7*, 253–264.
Richter, R., & Döllner, J. (2014). Concepts and techniques for integration, analysis and visualization of massive 3D point clouds. *Computers, Environment and Urban Systems*, *45*, 114–124.
Ropinski, T., & Hinrichs, K. (2005). *Interactive volume visualization techniques for subsurface data*. Visual information and information systems, 121–131.
Rossignac, J., Megahed, A., & Schneider, B. -O. (1992). Interactive inspection of solids: Cross-sections and interferences. *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 353–360). ACM.
Schwarz, M., & Seidel, H. -P. (2010). Fast parallel surface and solid voxelization on GPUs. *ACM Transactions on Graphics*, *29*, 1–10.
Shade, J., Gortler, S., He, L. -w., & Szeliski, R. (1998). Layered depth images. *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques* (pp. 231–242). ACM.
Shojaei, D., Kalantari, M., Bishop, I.D., Rajabifard, A., & Aien, A. (2013). Visualization requirements for 3D cadastral systems. *Computers, Environment and Urban Systems*, *41*, 39–54.
Thomas, T.A. (1968). *Technical illustration* (2nd ed.). New York: McGraw-Hill.
Trapp, M., Beesk, C., Pasewaldt, S., & Döllner, J. (2011). Interactive rendering techniques for highlighting in 3D geovirtual environments. In T.H. Kolbe, G. König, & C. Nagel (Eds.), *Advances in 3D geo-information sciences* (pp. 197–210). Berlin Heidelberg: Springer.
Trapp, M., & Döllner, J. (2008). Real-time volumetric tests using layered depth images. *Proceedings of Eurographics, 2008*.
Trapp, M., & Döllner, J. (2013). 2.5D clip-surfaces for technical visualization. *Journal of WSCG*, *21*, 89–96.
Trapp, M., Glander, T., & Buchholz, H. (2008). 3D generalization lenses for interactive focus + context visualization of virtual city models. *2010 14th International Conference Information Visualisation* (pp. 356–361). IEEE.
van Oosterom, P. (2013). Research and development in 3D cadastres. *Computers, Environment and Urban Systems*, *40*, 1–6.
van Oosterom, P., Stoter, J., & Lemmen, C. (2005). Modelling of 3D cadastral systems. In K. Kyu-Tae (Ed.), *Proceedings of the 28th Cadastral Seminar* (pp. 594–606). Busan: Korea Cadastral Survey Corp.
Vandysheva, N., Sapelnikov, S., Oosterom, P. J. M. V., Vries, M. E. D., Spiering, B., Wouters, R., et al. (2012). *The 3d cadastre prototype and pilot in the russian federation. The 3D Cadastre Prototype and Pilot in the Russian Federation*. Rome: FIG Working Week 2012.
Wang, S.W., & Kaufman, A.E. (1993). Volume sampled voxelization of geometric primitives. *Proceedings, IEEE Conference on Visualization, 1993. Visualization '93* (pp. 78–84).
Wang, C., Pouliot, J., & Hubert, F. (2012). Visualization principles in 3D cadastre: A first assessment of visual variables. *3rd International Workshop on 3D Cadastres: Developments and Practices. 25–26 October 2012, Shenzhen, China*. International Federation of Surveyors (FIG).
Westermann, R., Sommer, O., & Ertl, T. (1999). Decoupling polygon rendering from geometry using rasterization hardware. In D. Lischinski, & G. Larson (Eds.), *Rendering Techniques' 99* (pp. 45–56). Vienna: Springer.
X3D (2013). X3D architecture and base components V3, ISO/IEC 19775-1:2013. *Web3D consortium*.
Xie, X., Zhu, Q., Du, Z., Xu, W., & Zhang, Y. (2013). A semantics-constrained profiling approach to complex 3D city models. *Computers, Environment and Urban Systems*, *41*, 309–317.
Zhou, Y., Dao, T.H.D., Thill, J. -C., & Delmelle, E. (2015). Enhanced 3D visualization techniques in support of indoor location planning. *Computers, Environment and Urban Systems*, *50*, 15–29.
Zhu, Q., Zhao, J., Du, Z., Zhang, Y., Xu, W., Xie, X., et al. (2011). Towards semantic 3D city modeling and visual explorations. In T.H. Kolbe, G. König, & C. Nagel (Eds.), *Advances in 3D geo-information sciences* (pp. 275–294). Berlin Heidelberg: Springer.

## Web references

Soltesz, Frank (2014, Dec. 4thh). http://telstarlogistics.typepad.com/telstarlogistics/2007/03/index.html (Telstar Logistics® gave a brief introduction to Frank Soltesz in 2007. Dr. Chris Mullen has collected some of the most famous illustrative artworks by Frank Soltesz in his private website, http://www.fulltable.com/vts/c/cut/factories/a.htm, Dec. 4th, 2014).
HAER (Historic American Engineering Record) (1974). *Cross section looking west — Phoenix Mill, north bank of Still River, Phoenixville, Windham County, CT.* Register Number: HAER CONN,8-PHOE,1 (sheet 7 of 7). Washington, D.C. 20540 Washington, D.C.: USA Library of Congress Prints and Photographs Division (http://hdl.loc.gov/loc.pnp/pp.print. It is also available on Wikipedia Commons, http://commons.wikimedia.org/wiki/File:Cross_Section_Looking_West_-_Phoenix_Mill,_North_bank_of_Still_River,_Phoenixville,_Windham_County,_CT_HAER_CONN,8-PHOE,1-_(sheet_7_of_7).png).
Auctionata online© (2014, Dec. 4thh). Engraving, architectural view of Caprarola 1663. http://auctionata.com/intl/o/34543/engraving-architectural-view-of-the-villa-caprarola-1663?noredir