A Data Model for the Interactive Construction and Correction of 3D Building Geometry Based on Planar Half-Spaces

Martin Kada, Andreas Wichmann, Nina Manzke and Yevgeniya Filippovska

Abstract 3D city models of large areas can only be efficiently (re-)constructed using automatic approaches. But since there is always a certain number of buildings where the automation fails, there is a need for interactive construction and correction tools. These tools should ideally use the reconstruction results as input, so that the amount of manual labor is minimized. However, automatic 3D building reconstruction approaches make use of different solid modeling techniques that are not all suitable for interactive modeling purposes. One such representation is half-space modeling that exhibits several advantages for the automatic (re-)construction of 3D building models (from segmented point clouds). Because planar half-spaces are infinite entities that are usually represented as mathematical inequality equations, it is difficult to design an interactive modeling system that allows their direct manipulation. In this paper, we propose an interactive modeling concept specifically for 3D building geometry based on a half-space kernel. Following from it, a special-purpose object-oriented data model is developed that hides the kernel under a layer of parameterized primitives and boundary representation (B-rep) that give semantic meaning to building elements and is thus better comprehensible to human users.

M. Kada (✉) · A. Wichmann · Y. Filippovska
Institute of Geodesy and Geoinformation Science, Technische Universität Berlin, Berlin, Germany
e-mail: martin.kada@tu-berlin.de

A. Wichmann
e-mail: andreas.wichmann@tu-berlin.de

Y. Filippovska
e-mail: yevgeniya.filippovska@tu-berlin.de

N. Manzke
Institute for Geoinformatics and Remote Sensing, University of Osnabrück, Osnabrück, Germany
e-mail: nina.manzke@uni-osnabrueck.de

A. Abdul-Rahman (ed.), *Advances in 3D Geoinformation*,
Lecture Notes in Geoinformation and Cartography,
DOI 10.1007/978-3-319-25691-7_26

1 Introduction

In the last few years, the use of 3D city models has increased in the everyday life of people. For example, virtual models became recently popular for the visualization of 3D maps and virtual globes in car and pedestrian navigation systems. There is also a great demand for city models in mapping offices and in the real estate sector. Furthermore, they are required for applications such as noise and solar potential analysis, flood and pollutant simulations, and location planning of mobile antennas. In general, 3D city models include buildings, digital images of the urban area, and often roads, vegetation, and street furniture. For the reconstruction of 3D building models, several automatic approaches have been developed over the last two decades. An overview is, e.g., given in (Haala and Kada 2010).

The quality of such automatically reconstructed buildings is sufficient for most visualization purposes. However, for critical analyses and simulations, which require more accurate models, fully automatically reconstructed models might not be sufficient and need to be manually corrected. This process is usually very time consuming especially for large towns and cities, so that efficient modeling tools are needed. Therefore, we present in this paper an intuitive and interactive 3D modeling framework, which helps to avoid typical errors in the final building models and which supports the user during the construction process.

For the modeling of buildings, there already exist several solid modeling techniques. Popular methods are, e.g., boundary representation (B-rep), cell decomposition, and binary space partitioning tree (BSP-tree). However, they are not very intuitive to use for interactive modeling purposes, because in order to consider certain dependencies the underlying data structure has to be changed several times during the modeling process. Additionally, the operations which can be applied to the solid do not ensure the topological correctness of a model (e.g. its closeness) and the retention of geometric constraints (e.g. parallelism, rectangularity). Especially for a real-time application this can be implemented only with difficulty.

Modeling approaches belonging to the so called family of constructive modeling (Mäntylä 1988) better fulfill these requirements. For example, a widely used modeling technique in computer aided design (CAD) is constructive solid geometry (CSG). In CSG, a solid is built from a small number of predefined simple primitives that can be modified by affine transformations and combined with Boolean set operations. Using CSG automatically solves a number of issues. Amongst others, the Boolean set operations guarantee topological correctness of the resulting solids if the used primitives are already topologically correct. Additionally, CSG implicitly preserves in the resulting solid the geometric constraints of the primitives. If required, a CSG solid can be automatically and unambiguously converted to a B-rep, e.g. by the topology preserving Euler operators (Baumgart 1975).

In our modeling framework we use half-space modeling, which is a simplified form of CSG modeling. Each half-space is represented by a façade or a roof face that can be interactively modified by a user. The additional effort, which is accompanied by the use of half-space modeling, is in our framework hidden from

the user and is nevertheless suitable for the interactive 3D modeling in real-time. This concept allows therefore on the one hand a time efficient construction of new building models. On the other hand, existing models, which are obtained by an automatic reconstruction approach from an airborne captured point cloud as described, e.g., in (Kada and Wichmann 2013), can be corrected manually in a post-processing step.

2 Related Work

For the representation of 3D solids several modeling techniques have been developed (Hoffmann and Vaněček 1991). All these representations have in common that they organize in different ways the same geometric and topological data in form of a data structure. Generally, these data structures are based on the boundary, on the spatial subdivision, or on the operations of volumetric primitives that construct the solid (Mortenson 1985). According to Mäntylä (1988), there are three major types of data model schemes: boundary representation (B-rep) models, decomposition models, and constructive models.

A solid in B-rep is composed of its topological and geometrical information. It is represented as a collection of connected surfaces that defines the boundary between the solid and its complementary exterior according to the Jordan-Brouwer theorem. For an efficient access to adjacent and incident elements special data structures like Winged-Edge (Baumgart 1974), Half-Edge (Weiler 1985), Quad-Edge (Guibas and Stolfi 1985) have been introduced. In general, B-rep can represent a wide class of objects, but they are usually complex and additional effort is needed to guarantee the topological correctness of a solid and to maintain its geometric constraints. Therefore, this representation is usually more suitable for visualization rather than for interactive modeling purposes. The B-rep for the automatic reconstruction of buildings is, e.g., used in (Ameri and Fritsch 2000).

In a decomposition model, a collection of adjoining, nonintersecting simple primitives comprises the solid. One of its most general form is the cell decomposition model that defines a solid by a set of primitive cells. Due to its approximate nature, it is rarely used for visualization purposes but more frequently in analyses, e.g. by applying the finite element method. Some examples for the reconstruction of buildings based on cell decomposition and the BSP-tree are given in (Kada and McKinley 2009; Sohn et al. 2008).

Constructive models represent a solid as a combination of simple geometric shapes. One of its most common form is constructive solid geometry (CSG). It describes a solid as a hierarchical Boolean combination of primitives in terms of a tree. Thereby, primitives are stored in the leaf nodes and regularized Boolean set operations or rigid motions in the internal nodes (Mäntylä 1988). The evaluation of the tree always results in valid solids (as long as the primitives are also valid). Examples of CSG models in the reconstruction process of 3D buildings are given in (Brenner 2004; Kada and Wichmann 2013; Xiong et al. 2015).

The described data models are also used in interactive modeling systems. For example, several interactive mesh editing techniques based on B-rep models have been developed (Sorkine 2005). These techniques are generally suitable for the intuitive modification of freeform surfaces, but are not practically applicable to regularized polyhedral building models which have to preserve many geometric constraints. Another shape-modeling technique based on the concept of Lagrangian surface flow is used in (Duan et al. 2005). A real-time interactive visual editing of shape grammars for procedural architecture is shown in (Lipp et al. 2008). An interactive 3D solid modeling system for generating 3D models of architectural structures and urban scenes from unordered sets of photographs is given in (Sinha et al. 2008). Other well-known modeling systems (e.g. Cinema 4D, 3ds Max, AutoCAD, SketchUp, etc.) maintain several representation schemes of a solid and support the efficient conversion between them. They are, however, designed for modeling general shapes and not specifically designed to efficiently construct 3D building models.

3 3D Modeling Using Half-Spaces

In this section, the fundamentals of half-space modeling are summarized. For a detailed description, see, e.g., (Mäntylä 1988) or (Foley et al. 1990).

Half-space modeling is a form of solid modeling and belongs to the family of constructive modeling. It defines a solid as a combination of half-spaces obtained by Boolean set operations. A half-space H of $\mathbb{R}^3$ is a point set of the form $H = \{(x, y, z) \mid g(x, y, z) \leq 0\}$ where $g: \mathbb{R}^3 \rightarrow \mathbb{R}$ is its characteristic function. The subset of points $S_H = \{(x, y, z) \mid g(x, y, z) = 0\}$ is called the surface of the half-space H. The complementary half-space H^C to H is the half-space $H^C = \{(x, y, z) \mid g(x, y, z) \geq 0\} = \{(x, y, z) \mid (-g)(x, y, z) \leq 0\}$. Obviously, the intersection of half-space H and its complementary half-space H^C is the surface S_H. In our modeling framework only half-spaces with a linear characteristic function g are considered. These convex half-spaces are called planar half-spaces and their points, called interior points, satisfy the inequality

$$g(x, y, z) = Ax + By + Cz + D \leq 0 \tag{1}$$

where (A, B, C) is the normal unit vector of the plane $\{(x, y, z) \mid Ax + By + Cz + D = 0\}$ with the distance |D| to the origin.

Any polyhedral convex solid can be constructed by applying the Boolean set operation intersection to a set of planar half-spaces. Then, the solid points are just the interior points common to all of these half-spaces and each planar surface of the solid lies in the surface of a half-space. For example, a cuboid building can be represented as the intersection of six planar half-spaces. For the construction of a saddleback roof building only the half-space that limits the height of the flat roof has to be replaced by two sloped half-spaces as shown in Fig. 1.

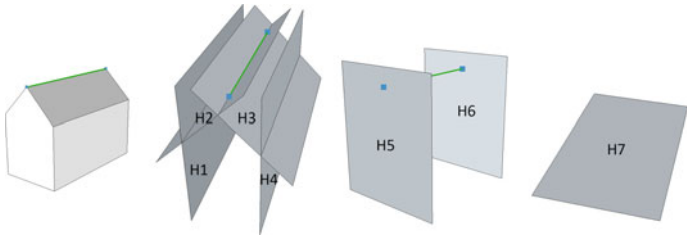

Fig. 1 A 3D building model with a saddleback roof defined by seven planar half-spaces

However, the construction of a non-convex solid is more difficult, because it cannot be modeled with the Boolean set operation intersection only. It therefore needs to be composed from convex components. This can be done since any non-convex solid can be decomposed into convex components. Then, the Boolean set operation union is applied to the set of convex components in order to construct the final non-convex solid. Therefore, half-space modeling is not restricted to certain geometric shape types.

Half-space modeling can conveniently be used in the automatic 3D building reconstruction from 3D point clouds, as buildings are commonly approximated by polyhedrons and there is a one-to-one correspondence between the segmented planar areas, the half-spaces, and the faces of the (convex) building blocks used to construct such models (see e.g. Kada and Wichmann 2013). Due to its mathematical formulation, the generated shapes are guaranteed to be topologically correct, although some care has to be taken to construct finite solids. There is also not as much interdependence in the definition of the half-space parameters as compared to the elements of a B-rep. For example, if a sloped half-space of a hip roof is changed, it has no effect on the other half-spaces, whereas changing the geometry of a face in B–rep results in further changes related to adjacent faces and incident edges and vertices. Also due to the use of Boolean set operations, geometric shapes can be generated by several loosely coupled components, e.g. by separating the model construction of the main building roof shape, the endings, the connections, and the outlines as described in Sect. 5. Such a flexibility is more complicated to implement using B-rep, because the faces, edges, and vertices have to be tediously calculated considering all shape combinations and parameter values.

4 Interactive Modeling Concept

We defined and formulated an interactive modeling concept that serves as a basis for the development of a data model that is suitable for our intended purposes. See Fig. 2 for a depiction of our concept.

The utilization of planar half-spaces in the (automatic) (re-)construction of 3D building models has several advantages. However, these come with the

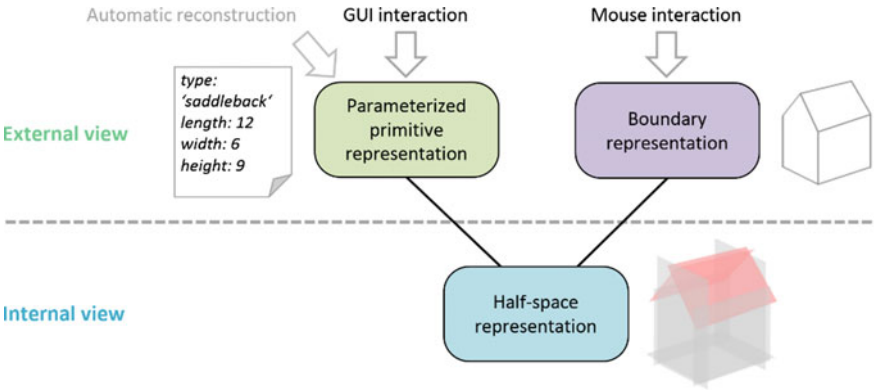

Fig. 2 The relation of the geometric representations within our interactive modeling concept

shortcoming that planar half-spaces are infinite entities represented as mathematical inequality equations. It is therefore very difficult to design an interactive modeling environment with a natural look and feel that allows their direct manipulation. So, the half-space entities as well as the entire half-space modeling functionality are in our concept encapsulated inside a kernel module and thus completely hidden from the user. This module is, on the one hand, as generic as possible to allow the construction of all common building shapes in existence and, on the other hand, it reflects our targeted interaction patterns and workflows to simplify and support the whole 3D building model construction process. Because of its invisibility towards the user, the kernel module can be regarded as the internal view of our data model.

On top of the kernel module, two further modules are specified that serve as the interface to the automatic 3D building reconstruction and to the user; thereby constituting the external view. They use different geometric representations, each one being best suited for their specific task, namely parameterized primitive representation and B-rep.

In the parameterized primitive representation module, the 3D building model construction process is presented to the reconstruction application and to the interactive user via a library of building blocks that feature common roof shapes. These primitives can be semantically parameterized, geometrically molded (up to a certain extent), freely placed in a building specific coordinate system, and combined with other primitives to form all possible 3D building shapes. In our context, we consider semantic parameters as being descriptive with regard to the building roof geometry and to be in most instances specific to certain building roof types (e.g. ridge length, eaves height, hip slope, etc.). Users can directly set and change these semantic parameters with common controls (buttons, spinners, list boxes, etc.) within a graphical user interface. From the primitive type information and the primitive parameter values, a unique half-space representation can be computed. Because a conversion is not practically feasible in the other direction without providing further semantic information, the parameter representation is internally

our basic representation for manipulations that is also used to store and exchange 3D building models via files.

Neither the parameterized primitives nor the half-space representation allows a graphical output that is needed for an interactive system. Real-time 3D visualization and rendering systems work (in their core) only with points, lines, and (triangulated) polygons. They cannot make use of models given as a set of parameters or mathematical half-space equations. For this reason, a B-rep is further needed that specifies a building model and its components as a collection of vertices, edges, and faces. The B-rep is not derived and constructed from the primitive parameters themselves, but by a conversion from the half-space representation to exploit the above mentioned advantages of half-space modeling. Besides rendering purposes, the graphical primitives also provide the means for human-machine interactions as they can be selected and moved in a 3D view with the help of a computer mouse. The modeling system then translates the user interactions to the internal parameterized primitive representation. In order to be able to identify the relevant parameters, the boundary elements are semantically linked to those half-spaces that they have been generated from. A face is associated with one half-space, an edge with two half-spaces, and a vertex with three or more half-spaces respectively. Most half-spaces have a meaning and their combination (in terms of intersections) form specific roof shape elements. For example, the intersection of the two sloped half-spaces of a saddleback roof form a special edge in the B-rep: the ridge line. If the ridge line is lowered or elevated, the system evaluates that it effects the ridge height of the primitive.

Automatic model-based 3D building reconstruction approaches as well as manual editors using a parameterized primitive representation usually offer a library of 3D primitives with rectangular footprints and standard roof shapes (saddleback, hip, mansard, etc.). Simple shaped buildings with no roof superstructures (dormers, chimneys, etc.) can then be constructed by a single primitive. The shape of most buildings are, however, more complex and need to be constructed by combining several primitives. Besides placing primitives and choosing roof shape parameters, the user also needs some means to determine the individual outline of each primitive.

Our proposed concept for an interactive 3D building model construction workflow consists of the following four steps that are repeated for each building primitive (cp. Fig. 3):

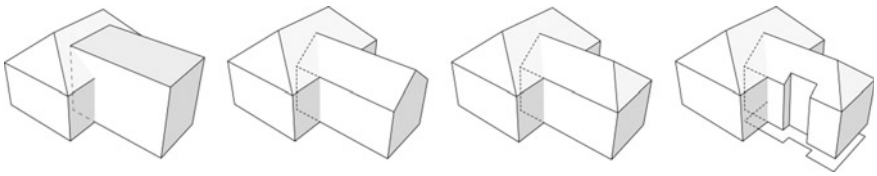

Fig. 3 The four steps of placing and defining the geometry of a parameterized building primitive in the described interactive 3D building modeling concept

1. A basic parameterized building primitive with cuboid shape is added to the 3D building model and selected as the current primitive. The position, orientation (with regard to the horizontal plane), and the extent (length, width, and height) of the primitive are set, preferably via mouse interactions.
2. The basic roof shape of the currently selected primitive is chosen from a list of available roof types (flat, shed, saddleback, hip, mansard, tip roof, etc.). Then the roof type specific parameters are set. For example, the position and the height of the ridge line of a saddleback roof as well as the two heights of the eaves lines.
3. For primitives of type ridge roof, the shape of the front and the back side is determined by choosing an ending type and setting ending specific parameters. Examples for common endings are gable, hip, broken hip, mansard hip, and Dutch gable. It should also be possible to interconnect primitives of this roof type with specific ending objects to form L- and T-connections.
4. For each primitive, a 2D polygon can be digitized that determines the detailed outline (and thereby the facades) of the primitive by cropping away those parts of the primitive that lie outside this polygon.

The user repeats these four steps until the constructed primitives reproduce the building according to the user's requirements. An example is given in Fig. 3.

5 Data Model for 3D Buildings

In this section, we describe our data model that combines half-space modeling and parameterized primitives to represent the 3D geometry of buildings. It is modeled in an object-oriented way and defines a hierarchy of classes (see Fig. 4). The half-space kernel serves as its foundation and describes the general structure of 3D building models. For a user oriented description of common roof shapes, a number of specialized classes for parameterized roof shapes, as well as endings and connections that are specific for ridge based roofs are further derived.

5.1 Half-Space Kernel

In our data model, every 3D building model consists of a collection of building components. These components are solids with flat side faces and common roof shapes. They are combined to form the basic shape of a building and also to model roof superstructures like dormers and chimneys. In terms of half-space modeling, a 3D building model is the Boolean union of n components:

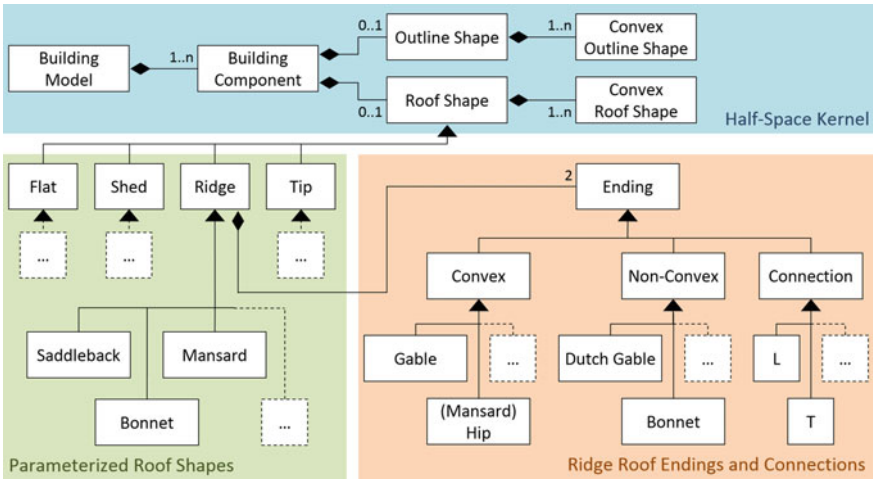

Fig. 4 The proposed data model for 3D building models

$$Building = \bigcup_{i=1\ldots n} Component_i \tag{2}$$

Initially, each component has a cuboid shape with parameterized extent (length, width, height). They are each defined in their local coordinate system and have position and orientation parameters that allow their transformation into the superior coordinate system of the building. The use of local coordinate systems greatly simplifies the definition of the components' half-space parameters and their interactive placement and rotation. The half-space representation of a cuboid shape is then defined as the intersection of six axis-aligned planar half-spaces:

$$Cuboid_i = \bigcap_{j=1\ldots 6} H_{ij} \tag{3}$$

As components with cuboid shape are not sufficient to construct realistic building models, it is necessary to be able to further develop their shape with regard to their roof and outline. So a component may have a roof shape object and an outline shape object associated with it. The overall shape of a component is then defined as the Boolean intersection of the cuboid, the roof, and the outline shape:

$$Component_i = Cuboid_i \cap RoofShape_i \cap OutlineShape_i \tag{4}$$

Because the roof shape and the outline shape as well as the components cuboid shape should all be independently defined of one another, they solely consist of only those half-spaces that are actually required to shape out their specific part of the component. Therefore, roof shapes consist of mainly sloped and horizontal half-spaces, whereas outline shapes consist only of vertical half-spaces. Both shapes are not bounded in all directions, which makes them semi-infinite solids. If no

half-spaces are provided, roof shapes and outline shapes are regarded as infinite sets. Only by their intersection with the cuboid, a finite solid component is generated.

As mentioned above, non-convex shapes are constructed as a Boolean union of convex shapes. So each roof shape and each outline shape is defined as a collection of convex roof shapes and a collection of convex outline shapes respectively:

$$RoofShape_i = \bigcup_{k=1\ldots n} ConvexRoofShape_{ik} \tag{5}$$

$$OutlineShape_i = \bigcup_{l=1\ldots n} ConvexOutlineShape_{il} \tag{6}$$

Convex roof shapes and convex outline shapes are, again, described by the intersection of half-spaces, analogous to the definition of the cuboid shape. A saddleback roof shape, e.g., is then defined by one convex roof shape that contains two sloped half-spaces. A bonnet roof shape, which has a non-convex shape, is defined by two overlaid convex roof shapes that each contains two sloped half-spaces (see e.g. Fig. 5).

The outline shape can be regarded to only effect the component in 2D. It crops away surplus regions of a component that are outside the outline shape like a cookie cutter. In fact, during the interactive model construction the outline is inputted by the user as a 2D polygon. However, since the outline shape is defined for a 3D model, its half-space representation must also be given in 3D. It consists of strictly vertical half-spaces and is therefore not bounded in the upward and downward direction. Non-convex outlines are given as the union of convex outline shapes. An example for an L-shaped outline is given in Fig. 6.

In order to maintain meaningful parameters, roof shapes should completely fill out the footprints of cuboids. Otherwise, e.g. when the slope of a half-space from a saddleback roof is too steep and portions of the rectangular footprint of the

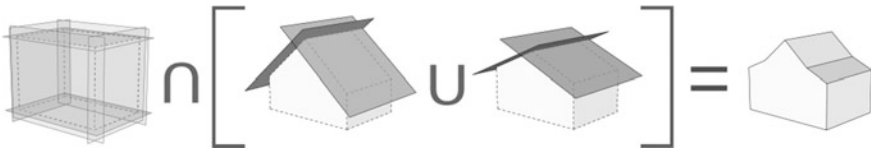

Fig. 5 The construction of a building component with bonnet roof shape using half-spaces

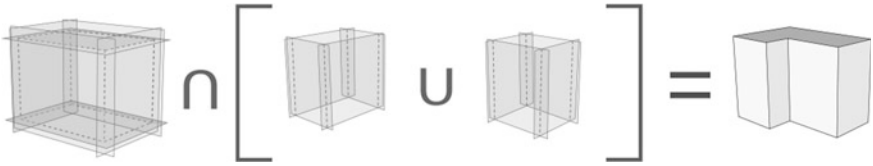

Fig. 6 The construction of an L-shaped building outline using half-spaces

component gets clipped away, then the shortened geometry of the component does not fit its semantic extent parameters anymore.

5.2 Parameterized Roof Shapes

As described in the interactive modeling concept, the parameterized roof shape module hides the half-space representation from the user. So from the roof shape class of the half-space kernel module, four base classes are derived that assign every roof shape to the types flat, shed, ridge, and tip roof. From these base classes further specialization classes are derived. Their main purpose is to semantically restrict the types of roof shapes that can be constructed and to introduce meaningful roof shape parameters. As the extent parameters of a component define the limiting geometry of each specialized roof shape, they are incorporated in the parameters of the derived specialization classes. For example, the height parameter of the component represents the maximum height of each roof shape and is identical to the ridge or tip height of a saddleback roof or tower roof respectively. From the roof shape parameters, the roof shape objects are constructed and the half-spaces are derived according to the half-space kernel.

The ridge roof shape is of particularly interest here, because building components with this roof type are often interconnected and their modeling is therefore interdependent. From the ridge roof shape class, all specialization classes for roof shapes that feature a ridge are derived. Ridge roof shapes may have two ending objects associated with them. These are used to further define the two sides of the roof shape where the ridge line ends or where two (or more) components with ridge roof shapes interconnect. This is explained in more detail in the next subsection.

Other specialized roof shape classes with flat, shed, and tip roof type can be defined accordingly. Building components with specialized roof shapes can not only be used as basic building blocks for the 3D building shape itself, but also for the construction of dormers, chimneys, towers and other roof elements.

5.3 Ridge Roof Endings and Connections

Each object of type ridge roof shape has two ridge roof ending objects associated with it that allow to further form the front and back side of this particular roof shape class. Without ending objects, ridge roof shapes are only delimited by the two vertical half-spaces from the basic cuboid component; resulting in two gables. The ending concept allows to support further ridge roof ending shapes that are semantically distinguishable and that provide their own geometric embodiment. The two ending objects E_1 and E_2 define, on the one hand, two sets of half-spaces E_1H and E_2H that form convex shapes and that can therefore be directly intersected with the roof shape. In order to construct non-convex ending shapes for both

endings, the ending objects may, on the other hand, also define two sets of convex solids E_1S and E_2S that must be combined with the roof shape using the Boolean union operation. This results in an extended definition of the roof shape formula:

$$RoofShape_i = \left(\left(\bigcup_{k=1\ldots n} ConvexRoofShape_{ik} \right) \cap E_1H \cap E_2H \right) \cup E_1S \cup E_2S \quad (7)$$

Besides the base ending class, the data model defines three intermediate classes to distinguish between convex, non-convex, and connection endings. Further specialized ending classes are derived from these classes according to their geometric characteristic and purpose.

Ending objects of type convex are rather simple to define and to implement because they only specify a set of half-spaces that form a convex shape, but not a convex solid. Prominent examples of convex endings are gable, hip, and mansard hip. Because the geometry of a gable ending is already fully defined by the cuboid component in combination with a ridge roof shape, the gable ending class defines no further geometry and can be regarded as the standard ridge roof ending. The hip ending class defines one sloped half-space that is intersected with the ridge roof shape of the component. Fig. 7 depicts a building component with a saddleback roof shape that is associated with a hip ending. The advantage of half-space modeling becomes apparent in this example, because the definition of the hip ending is independent of the component's roof shape. The hip ending object could be associated, e.g., with a barrel or bonnet roof to have the same effect as on the saddleback roof. Other convex ending classes are defined analogously and may provide further half-spaces to form more complex ending shapes.

The intersection of ridge roof shapes with convex ending shapes is not expressive enough to define all kinds of standard roof shape endings. For example, non-convex endings like the Dutch gable roof cannot be constructed in this way. But the intersection with a vertical half-space, e.g., creates the necessary space within the cuboid building component that is required to construct the non-convex part of this particular roof (cp. Fig. 8). The missing front part is then constructed as a convex solid that is adjoined to the roof shape object by a Boolean union operation. Non-convex roof endings can define any number of convex parts to form any roof ending shapes in existence.

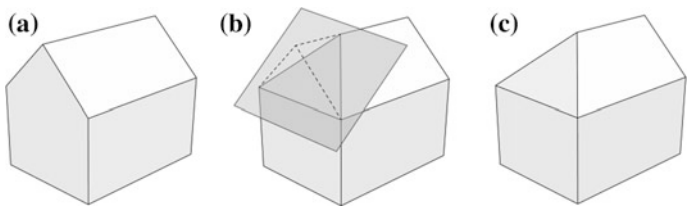

Fig. 7 A building component with a saddleback roof shape (**a**) clipped by the half-space of a hip ending (**b**) resulting in a building component with an asymmetric hip roof shape (**c**)

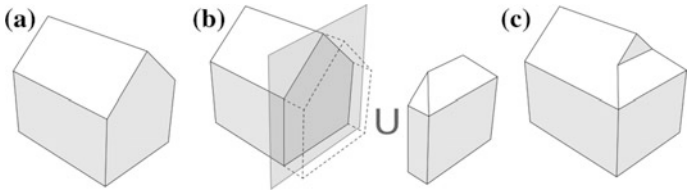

Fig. 8 A building component with a saddleback roof shape (**a**) clipped by the vertical half-space of a Dutch gable ending makes room for the front solid (that is attached to the component by a Boolean union operation) (**b**) resulting in a building component with an asymmetric Dutch gable roof shape (**c**)

For complex building shapes, it is often not sufficient to model building components independently from one another. The components rather need to be interconnected to form common shapes like L-, T- and X-connections. Endings of the derived type connection are a convenient way to accomplish this for components with a ridge roof shape.

Connection objects are special in regard to how they provide their half-spaces, because they reuse some of the half-spaces of one component for the refinement of the shape of the other component and vice versa. So the connection objects facilitate an information exchange in the form of half-spaces to simplify the model construction process. For example, to join two components with saddleback roof shapes in an L-connection, the connection adds the sloped half-space from the right component to the half-spaces of the left component to clip the overhanging saddleback roof and vice versa (see Fig. 9). It also exchanges the façade half-spaces in the same way to clip overhanging component parts in cases where the components form an acute angle.

The advantage of reusing half-spaces is not just the reduction of their overall number, but also that one component is automatically adapted if the other component changes. For example, if the ridge height of one component changes, then only the half-space parameters of the same component need to be updated; the half-space parameters of the other component remain unaffected.

Although we have only introduced two connection types (see Fig. 4), further types like the X-connection might be useful. Some shapes like the U-connection, however, are not necessary as they can be modeled by several L-connections.

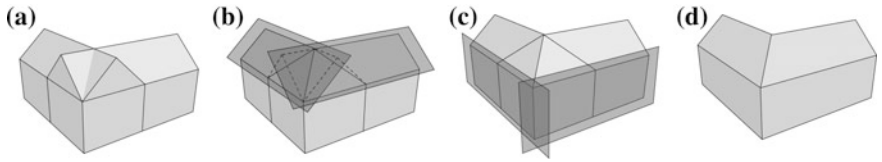

Fig. 9 Two building components with saddleback roof shape (**a**) that are clipped by the sloped roof shape half-spaces of the other component (**b**) as well as by the façade half-spaces (**c**) resulting in an L-shaped building model with ridge roof shape (**d**)

6 Implementation and Discussion

We have implemented the described data model and the prototype of an interactive system for constructing and correcting 3D building models. It is written in C ++, uses Qt and OpenGL for the GUI and the 3D rendering part as well as the commercial CAD kernel 3D ACIS Modeler from Spatial for the Boolean evaluation of half-spaces and their conversion to a B-rep.

In an interactive 3D modeling system, the real-time behavior is always of major concern. As the data model uses three representations for one and the same geometric entity, there is a continuous conversion process happening in the application's background, especially during the actual user interactions. Because the user typically modifies the geometry of only one building component at the time, the evaluation of its half-space representation and its conversion to B-rep is fast enough for real-time rendering purposes. In our implementation, an instantaneous feedback could still be observed on a standard desktop computer system even during the simultaneous interaction with about ten to fifteen components. To reduce the number of geometric computations, the rendering system could also be utilized, e.g. to perform the affine transformations during the interactive placement and rotation of component groups.

We have also implemented a number of validity checks for parameter values in our interactive environment that are executed in the background during user interactions to prohibit the construction of invalid building components. These are necessary to avoid, e.g., degenerated shapes.

Practical experiments with the systems showed that it satisfies its intended purpose and that the half-space modeling kernel could successfully be hidden from the user without restricting any functionality.

7 Conclusion and Future Work

In this paper, we have presented an interactive modeling concept for the construction and correction of 3D building models in cases where automatic methods fail. Here, it is assumed that the automatic reconstruction approaches generate solids using half-space modeling, which is not suited for manual editing. Therefore, a data model has been proposed with a half-space kernel that is hidden by a layer of parameterized primitives and B-rep. We have shown how half-space modeling can be used to construct the 3D building geometry by loosely coupled components like roof shapes, endings, and connections.

We conducted experiments to restrict the movement of B-rep elements in the 3D view. Hard coding restrictions into the application is, however, tedious, time-consuming, and not very flexible with respect to adding further roof and ending shapes. Therefore an automatic generation of restrictions for faces, edges,

and vertices of the B-rep is envisioned that results from the combination of basic restrictions on half-spaces. For example, the restrictions of an edge could be derived from basic restrictions of the two half-spaces it was generated from.

Acknowledgments The research work presented in this paper is supported by the German Research Foundation (Deutsche Forschungsgemeinschaft) under grand number KA 4027/1-1.

References

Ameri, B., Fritsch, D. (2000). Automatic 3D building reconstruction using plane-roof structures. In *Proceedings Of The American Society Of Photogrammetry And Remote Sensing Confere*nce, Washington, D.C.

Baumgart, B. G. (1974). *Geometric Modeling for Computer Vision*, Ph.D. thesis, Stanford University.

Baumgart, B. G. (1975). A polyhedron representation for computer vision. In *Proceedings of the AFIPS conference*, (vol. 44, pp. 589–596).

Brenner, C. (2004). Modelling 3D objects using weak CSG primitives. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science, 35*(3/B3), 1085–1090.

Duan, Y., Hua, J., & Qin, H. (2005). Interactive shape modeling using lagrangian surface flow. *The Visual Computer, 21*(5), 279–288.

Foley, J., van Dam, A., Feiner, S., & Hughes, J. (1990). *Computer Graphics: Principles and Practice* (2nd ed.). Reading, Massachusetts: Addison-Wesley.

Guibas, L., & Stolfi, J. (1985). Primitives for the manipulation of general subdivisions and the computation of voronoi diagrams. *Transactions on Graphics, 4*(2), 74–123.

Haala, N., & Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing, 65*(6), 570–580.

Hoffmann, C. M., & Vaněček, G. (1991). Fundamental techniques for geometric and solid modeling. In C. T. Leondes (Ed.), *Advances in Control and Dynamics* (pp. 101–165). : Academic Press.

Kada, M., & McKinley, L. (2009). 3D building reconstruction from lidar based on a cell decomposition approach. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science, 38*(3/W4), 47–52.

Kada, M., & Wichmann, A. (2013). Feature-driven 3D building modeling using planar halfspaces. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, II-3*, 189–196.

Lipp, M., Wonka, P., & Wimmer, M. (2008). Interactive visual editing of grammars for procedural architecture. *ACM Transactions on Graphics, 27*(3), 1–10.

Mäntylä, M. (1988). *An Introduction to Solid Modeling*. Rockville, Maryland: Computer Science Press.

Mortenson, M. (1985). *Geometric Modeling*. New York: John Wiley & Sons.

Sinha Sudipta, N., Steedly, D., Szeliski, R., Agrawala, M., & Pollefeys, M. (2008). Interactive 3D architectural modeling from unordered photo collections. *ACM Transactions on Graphics, 27* (5), 1–10.

Sohn, G., Huang, X., & Tao, V. (2008). Using a binary space partitioning tree for reconstructing polyhedral building models from airborne LIDAR data. *Photogrammetric Engineering and Remote Sensing, 74*(11), 1425–1438.

Sorkine, O. (2005). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing* (pp. 175–184).

Weiler, K. (1985). Edge-based data structures for solid modeling. *Curved-Surface Environments, IEEE Computer Graphics & Applications, 5*(1), 21–40.

Xiong, B., Jancosek, M., Oude Elberink, S., & Vosselman, G. (2015). flexible building primitives for 3D building modeling. *ISPRS Journal of Photogrammetry and Remote Sensing, 101*, 275–290.