

# Constructive Roofs from Solid Building Primitives

Johannes Edelsbrunner<sup>1(✉)</sup>, Ulrich Krispel<sup>1,2</sup>, Sven Havemann<sup>1</sup>,  
Alexei Sourin<sup>3</sup>, and Dieter W. Fellner<sup>1,4</sup>

<sup>1</sup> Institute for Computer Graphics and Knowledge Visualization (CGV),  
Graz University of Technology, Graz, Austria  
`{j.edelsbrunner,s.havemann}@cgv.tugraz.at`

<sup>2</sup> Fraunhofer Austria Research GmbH, Graz, Austria  
`ulrich.krispel@fraunhofer.at`

<sup>3</sup> School of Computer Engineering,  
Nanyang Technological University, Singapore, Singapore  
`assourin@ntu.edu.sg`

<sup>4</sup> GRIS, TU Darmstadt & Fraunhofer IGD, Darmstadt, Germany  
`d.fellner@igd.fraunhofer.de`

**Abstract.** The creation of building models has high importance, due to the demand for detailed buildings in virtual worlds, games, movies and geo information systems. Due to the high complexity of such models, especially in the urban context, their creation is often very demanding in resources. Procedural methods have been introduced to lessen these costs, and allow to specify a building (or a class of buildings) by a higher level approach, and leave the geometry generation to the system. While these systems allow to specify buildings in immense detail, roofs still pose a problem. Fully automatic roof generation algorithms might not yield desired results (especially for reconstruction purposes), and complete manual specification can get very tedious due to complex geometric configurations. We present a new method for an abstract building specification, that allows to specify complex buildings from simpler parts with an emphasis on assisting the blending of roofs.

**Keywords:** 3D-modeling · Procedural modeling · Architectural models

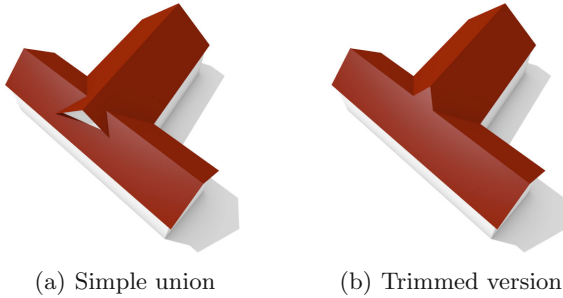
## 1 Introduction

With the growing popularity of virtual 3D worlds in games and movies, various VR simulations and 3D street walk-throughs of today's cities or cultural heritage scenes, attention of researchers shifts towards quick and flexible ways of modeling or reconstructing large numbers of buildings.

While manually generated models typically yield the highest visual quality, the effort to produce such results is immense. The research addressed at this problem can be grouped into two approaches: automatic reconstruction from measurement data, and automatic generation using procedural modeling.



**Fig. 1.** Modeling a building by parametrizable parts greatly reduces the necessary effort. A simple union of parts might lead to undesirable results (left). Therefore, we introduce automatic roof *trimming* for solid building primitives. The resulting model can be further refined using existing procedural approaches (right).



**Fig. 2.** A simple union of building parts with roof does not produce a desirable result (a). The influence of parts can be non-local, in this example the bounding plane of the roof geometry of the smaller building trims the roof geometry of the higher building and induces a small triangular roof area (b).

While automatic reconstruction might produce a faithful appearance of the original object, the result is often a (possibly dense) 3D mesh. This results in serious limitations to the modifiability of such models, as manual effort is required to change such models.

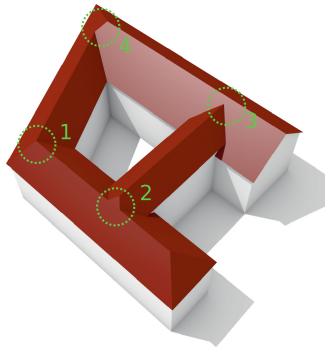
On the contrary, procedural modeling is an abstract representation of the building process of a model. Changing the procedural description typically requires less effort than manual 3D modeling, however the procedural system has to keep the balance between automation (lessen manual effort) and expressiveness of the system (which types of models can be created).

Current procedural modeling methods use mass modeling to define the coarse outline of a building by representing it via a number of parts. Rules further refine the geometry of these parts. Basic interaction between parts is possible via occlusion queries. Automatic roof synthesis can be done, but does not reflect all possible roof configurations present in real buildings.

Especially buildings in historic cities can often be modeled by simple parts, but achieving the final roof geometry is not possible by simple nesting or boolean union of the parts. The problem here is that the roof of one part can be influenced by the roof of another part (see Figs. 1 and 2). Depending on the constellation

of the parts, different topological connections arise (see Fig. 3). These problems can increase with growing irregularity of the building. Figure 4 shows a view of the city of Graz, Austria. Merging and influencing roof parts form a complex roof landscape. Still, blending of roof faces follows some consistent rules which we formalize in this paper.

The goal of this paper is to present a method that will allow representing a big variety of coarse building structures with roofs using a concise declarative approach. The structure is modeled by parts and their geometric influence to each other. The resulting structure can be input for a rule-based system that refines the geometry of the building parts. We show an example of both techniques combined in Figs. 1 and 21.



**Fig. 3.** Complex roof shapes. At (1, 2, 3, 4) the geometry has different topologies, depending on how roof-parts merge together. At (1) four roof-faces meet in one vertex, which is a special constraint since in general four planes do not intersect in one point.

We make the following Research contributions in this paper:

1. We present an *abstract building model* specification, that facilitates a concise description of a building assembled of several parts.
2. We introduce a method for automatic geometric trimming of adjacent building parts that influence each other.

## 2 Related Work

Shape grammars have been introduced by Stiny et al. [22] in order to generate paintings using rule systems. In [29] the concept was extended to split grammars in order to model 3-dimensional structures, especially building facades. The concept was further refined by [21]. Several other works were built on this principles, extending it to interconnected structures [14], extended systems of rule application [15], or more general split rules [27]. These systems can use the aforementioned mass modeling approach to generate rough geometry. While



**Fig. 4.** Aerial image of the inner city of Graz, Austria. Different roof parts merge and form a complex roof landscape. There are many implicit dependencies of parameters of roof faces like slope, height of eave, etc. This roof landscape is not easily created with existing modeling tools. Imagery ©2015 Google, Map data ©2015 Google.

most of these systems support basic roof generation, complex roof structures are either fully automatic (using a straight skeleton approach), limiting the number of possible roof structures, or the roof structures have to be modeled in detail for each part, resulting in a complex description.

3D construction of building roofs is an often covered topic in literature. Most papers are however concerned with reconstruction of roofs from image or scan data [2, 9, 13, 25, 28]. A good overview is given in [8]. The aforementioned methods usually use variants of plane fitting. The work of Milde et al. [19, 20] and Dorschlag et al. [4] use additional grammar and graph based approaches to further aid the reconstruction process. The method presented by Fischer et al. [7] uses an approach with connectors to align the extracted planes.

While they can produce good results, the output is often only a polygon mesh without semantic information. This is not very suitable for scenarios where a modifiable model is needed (e.g., urban planning), which is why recent methods use primitive fitting (with simple convex houses and gabled, hipped, flat, etc. roofs) where the primitives could be seen as semantic units [10, 11].

When models of non-existing objects have to be built (e.g., for movies, video games, etc.) none of the mentioned methods is applicable. There are many papers which deal with the 3D construction of buildings, however, the roof is most of the time only a small part of the solution and often modeled in a primitive and simple way. For example,

- [5, 6, 17, 21, 23, 24] use simple roof-primitives with often convex ground shape plans and gabled, hipped, flat, etc. roofs, and combine them to generate their roofs,
- [26] has a special specification for Asian roofs,
- [16, 18] use the famous straight skeleton algorithm [1] to generate the roof.

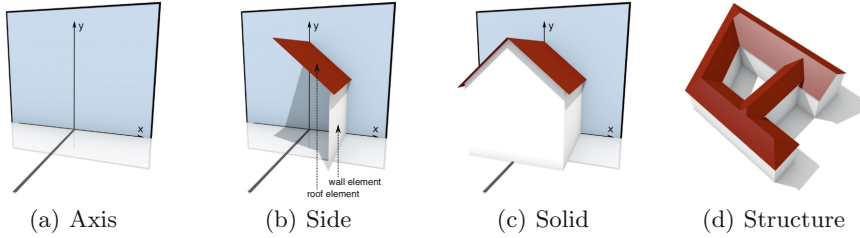
The methods with simple roof-primitives can yield good results for very regular buildings, but when the roof gets more complex and irregular, these methods are not able to fully reproduce the shape. Special specifications, like in the case for Asian roofs, are very suitable for the domain, but lack the possibility to describe a broad spectrum of roofs.

The straight skeleton algorithm is very suitable for generating general roofs, as its resulting skeleton corresponds to the edges of a roof on the building. Using it, a roof on an arbitrary simple ground polygon can be created. The downside is that the generated roof is only one of many possible roofs, and there is no way to get another roof. Here, an extension to the straight skeleton, called the weighted straight skeleton [3], brings greater flexibility. The work of Kelly et al. [12] uses the weighted straight skeleton to model not only arbitrary roof-shapes, but also the whole outer shell of a building. One drawback of this method is that straight skeleton algorithms (even more in the weighted case) are hard to implement due to algorithmic problems when numerical errors arise.

The problem of roof modeling also arises in geographic information systems. Open street map<sup>1</sup> is a project that lets volunteers all over the world collaborate in creating mapping data for streets, buildings, borders, etc. In order to model roofs, they have built a large categorization table of roofs. Each category has its own set of parameters. So a large class of buildings can be declaratively modeled by specifying the category and the corresponding parameters. Also, the straight skeleton method is contained in the categorization. Additionally, a method proposed for open street map is roof-line modeling. Here, the modeler traces roof lines on an aerial image (from top-down view) of the building and provides additional information, e.g. a type (ridge, edge, apex, ...), or height.

What is mostly missing is the accurate handling of merging roof faces of neighboring houses. In historic cities, houses that are adjacent often have the same roof. Depending on how the houses connect, the roof parts merge in different ways. While automatic reconstruction via plane fitting might reproduce the situation correctly (depending how good the algorithm is), methods that work with mass modeling primitives have difficulties, because for the merging the primitives must be changed. The weighted straight skeleton is capable of modeling these connections, but it might become unhandy when big roof regions with multiple roof part connections are modeled, because the whole roof must be modeled as one piece to ensure correct geometry at interconnections.

<sup>1</sup> <http://www.openstreetmap.org>.



**Fig. 5.** The components of our abstract building model. An *axis* is given for orientation. Relative to an axis, a *side* is defined. A side has wall and roof elements. Multiple sides form a *solid*, which is the basic building block for a *structure*.

### 3 Solid Building Primitives

In this section we give an overview of the building abstraction, and specify the individual components of our system.

The core abstraction in our system is the assembling of a building by separately defined parts, which are called *solids*. As most buildings are composed of planar walls, each solid is composed of several planar *side* parts, which represent a cross-sectional profile of the corresponding wall and roof part. Each side part corresponds to a line segment of the ground polygon of a solid. The geometry of a solid is obtained by intersection of half-spaces that correspond to the profile line segments of each side part.

A simple union of solids defined in this way might lead to undesirable results (see Fig. 2). However, we have observed that in many situations the correct result can be obtained by trimming solids by corresponding half-spaces of side parts of adjacent solids.

#### 3.1 Basic Components

Our abstract building model is composed of several components. We will now give an explanation of the basic components in a bottom-up manner (see Fig. 5):

##### Axis

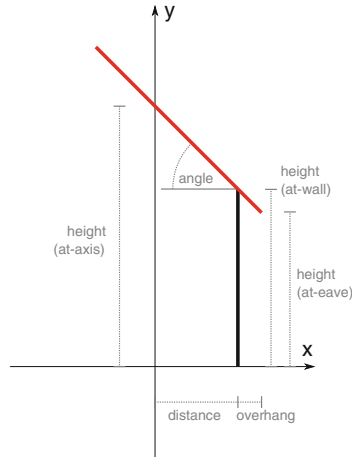
is a directed line segment in the ground plane, defined by a start- and an endpoint.

##### Side

corresponds to a planar building part consisting of a wall and a roof element. These elements are specified as line segments of the cross-sectional profile of the part, with respect to a reference axis that corresponds to the direction of a line segment of the ground polygon. Various parameters define the shape of the side (see Fig. 6).

##### Solid

is composed of multiple side components whose wall elements correspond to a convex ground polygon. Each line segment of the cross-sectional profile of a side corresponds to a half-space in 3D.



**Fig. 6.** Cross section view of a side. A side in our abstract building model consists of wall and roof elements. Here we show one possible parameterization (parameters in gray) from which the wall (black) and roof (red) elements are derived (Color figure online).

## Structure

is the combination (grouping) of several solids. Structures can be nested, e.g. for representing dormers.

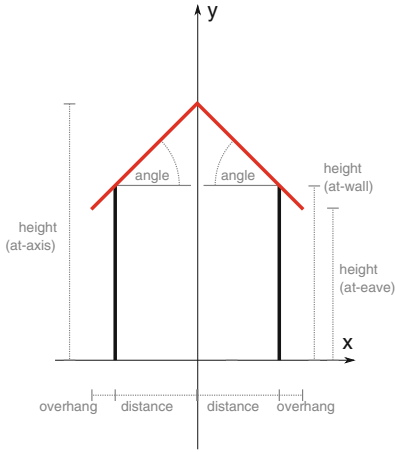
### 3.2 Parametrized Sides and Solids

Our definition of the side component allows to model a rich variation of different roof types and building parts. It is however relatively low-level, therefore we introduce parametrizations for sides and solids, that allow us to reduce the necessary effort to describe a building that is composed of similar parts. For example, if a building is modeled from aerial photographs, it is practical to define sides relative to ridge lines of roofs. When using ground polygons from a GIS (geographic information system) database, sides should be relative to the ground polygon. Different parametrizations for sides can be seen in Fig. 7.

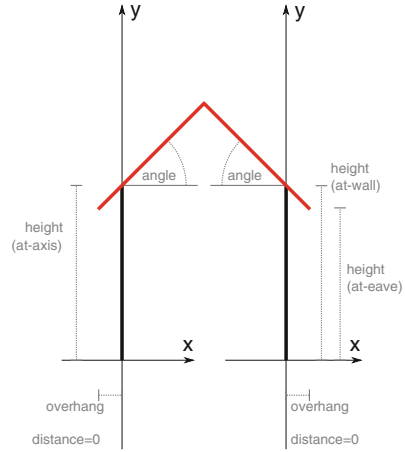
The parametrization includes an *overhang*, as roofs often extend over walls in order to prevent rain falling on walls.

### 3.3 Automatic Generation of Sides for Solids

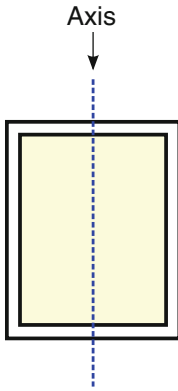
As described, the user can specify axis in the system. This can be done i.e. by tracing of aerial images. Subsequently a side is assigned to this axis. In order to reduce the amount of work needed, all sides of a solid can automatically be generated from this one given side. In the standard case of a rectangular solid this



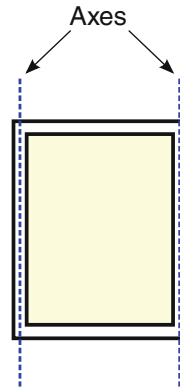
(a) Side parameters defined relative to one axis.



(b) Side parameters defined relative to two axes.



(c) Floor plan



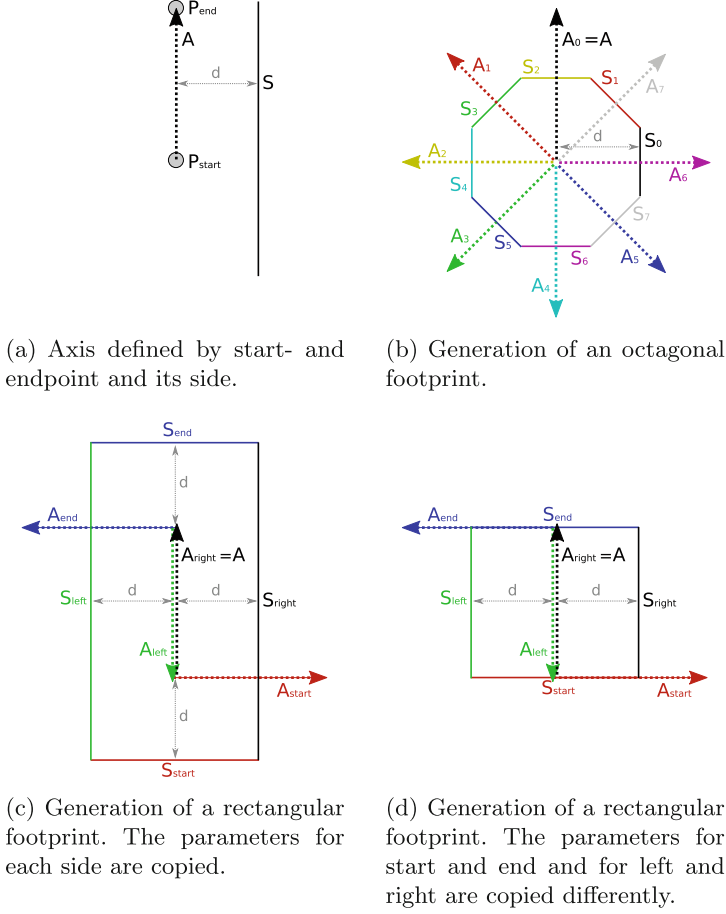
(d) Floor plan

**Fig. 7.** A solid is composed of multiple *side* components which can be parametrized arbitrarily. In (a, c) *side* components are specified with respect to the ridge (both *side* components use the same reference-axis and coordinate-system). (b, d) *side* components are specified with respect to the walls (*side* components use different reference-axes and coordinate-systems).

would be four sides generated from one given side, but also n-gon constellations are possible (see Fig. 8).

The parameters from the original side are automatically copied to the newly generated sides. But this process can also be modified. For example, when the ridge of a hipped house is traced, the copying of all parameters is useful since the roof has the same properties on all sides (slope, eave height, etc.) (see Fig. 8(c)). However, for a gabled house the start and end sides are different (see Fig. 8(c)). Here we tell the generation process to change the parameters for the start and





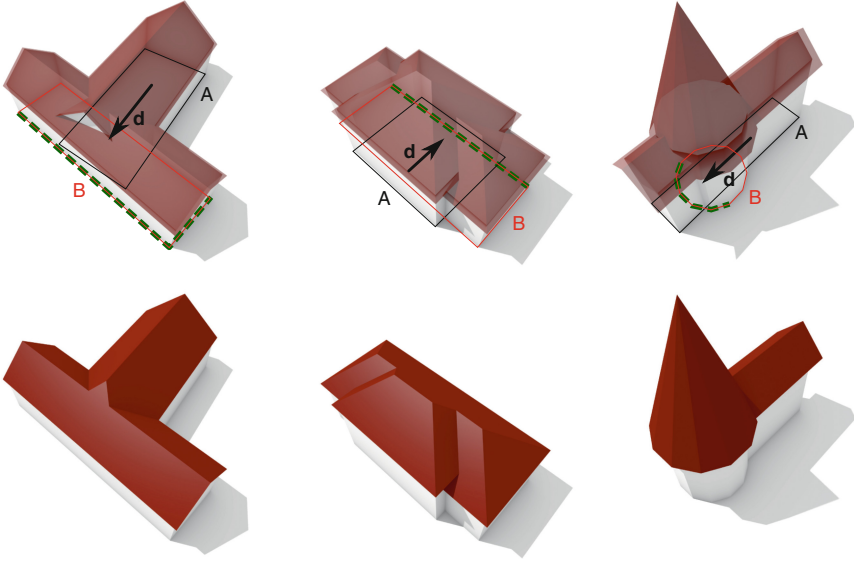
**Fig. 8.** Automatic generation of sides for solids. Based on an axis  $A$ , given by two points  $P_{start}$  and  $P_{end}$  and its according side  $S$  (a) additional sides for a complete solid can be generated. The axis is denoted  $A_i$  and the according sides  $S_i$ . Each side has parameters, the distance is here denoted with  $d$ . Different copy mechanisms allow for shapes in rectangular (c, d) or circular arrangement (b). The original parameters of the side are either copied for all parts (b, c) or modified while copying (d).

end sides. The distance is set to 0, and the roof components of this sides are removed.

### 3.4 Automatic Trimming of Solids

When multiple *solids* are combined using a simple union, unwanted configurations can occur like in Fig. 2(a). Therefore, we introduce *trimming* between solids: the geometry of one solid should be truncated by parts of the geometry

of another solid. In order to specify which parts get trimmed, a direction has to be specified.

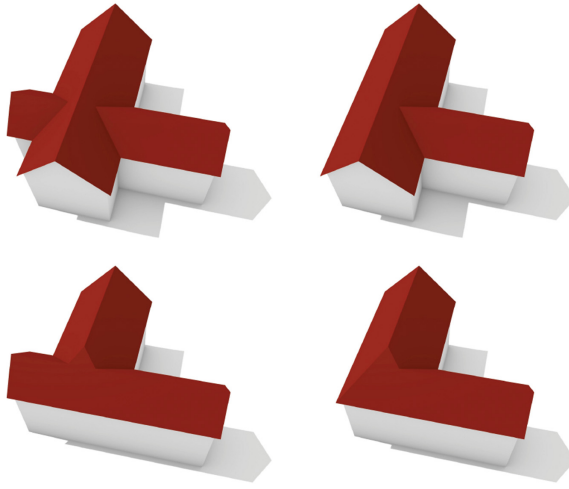


**Fig. 9.** We demonstrate the concept of trimming for solids by interactions of different building parts (top row). The user specifies an influence direction for each connecting solid (black arrow), the system automatically identifies the sides which will influence the result geometrically (green dashed lines on the ground polygon). Only sides corresponding to green segments will be used for the trimming (bottom row) (Color figure online).

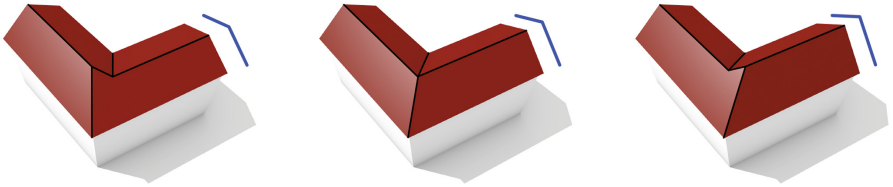
Recall that each side of a solid consists of wall and roof elements, which correspond to half-spaces in 3D. Assume having two solids  $A$  and  $B$ . We specify that  $A$  gets trimmed by  $B$  according to the direction  $d$  (see Fig. 9). Then, for each side  $s$  of  $B$  there is a test whether the outward facing normal vector of the wall component of  $s$  and  $d$  form a positive scalar-product. If yes (green segments in Fig. 9), the wall sub-solid of  $A$  is intersected by the half-space formed by the wall component of  $s$  and the roof sub-solid is intersected by the half-space formed by the roof component of  $s$ .

A trimming connection is directed, as one solid gets trimmed by another. Therefore, two solids yield four different possibilities of trimming variations between each other (see Fig. 10).

Using this method of trimming we can ensure that the influence between solids adapts correctly to changes in the parametrization. As an example, Fig. 11 shows the results of automatic trimming under varying roof angles for one solid.



**Fig. 10.** Two connected solids A and B can influence each other in four possible ways: No trimming (top left), A trimming B (top right), B trimming A (bottom left), and both solids trimming each other (bottom right). Note that different combinations would be possible by changing the influence directions.

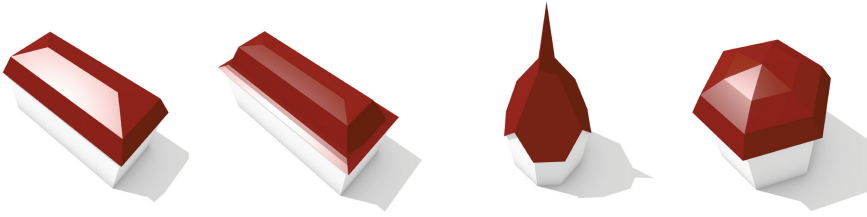


**Fig. 11.** A building that consists of two solids is evaluated with different parametrizations of the sides. In this example, only the slope of the roof of the right solid was gradually incremented from the first to the last image and the model was re-evaluated. It can be observed that the roof geometry adapts accordingly to the situation.

### 3.5 Roof Shapes

Up to now, we assumed the roof element of a side to be a single planar element. However, Mansard roofs have two or more slopes on their sides. We account for this by introducing side parametrizations that generate more than one roof element, which is called *profile-polygon* component in our specification. We show examples of a few roofs with profile polygons in Fig. 12.

Automatic trimming is performed in the same manner, although the case in which neighboring solids influence each other in a way that introduces additional geometry (compare to the small triangular area that emerges in Fig. 2(a)) might not be well defined. This, however, is not of practical importance, as we did not encounter such situations in real roof configurations, where these roofs usually have the same ridge height.



**Fig. 12.** Some roof shapes consist of more than one planar element, e.g. mansard roofs. We account for such types by a general profile-polygon type. The last two examples show solids with non-rectangular ground polygons.

## 4 Specification of the Building

In order to model a building we have to parameterize and combine multiple solids. We developed a specification with which this can be done in a structured fashion for one building.

### 4.1 Structure of Specification

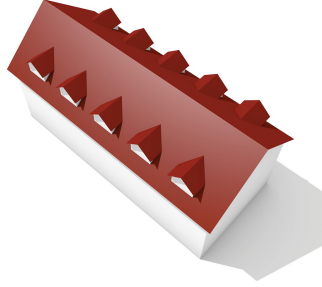
Figure 14 shows the elements of a structuring language we have developed. A building is represented as a structure, which consists of multiple solids. Each solid has at least one side, and a side has a parametrization (as already covered in Sect. 3.1).

To achieve the correct influence of solids as described in Sect. 3.4, a solid can contain multiple trimmings. A trimming references to the influencing solid and an axis that specifies the direction.

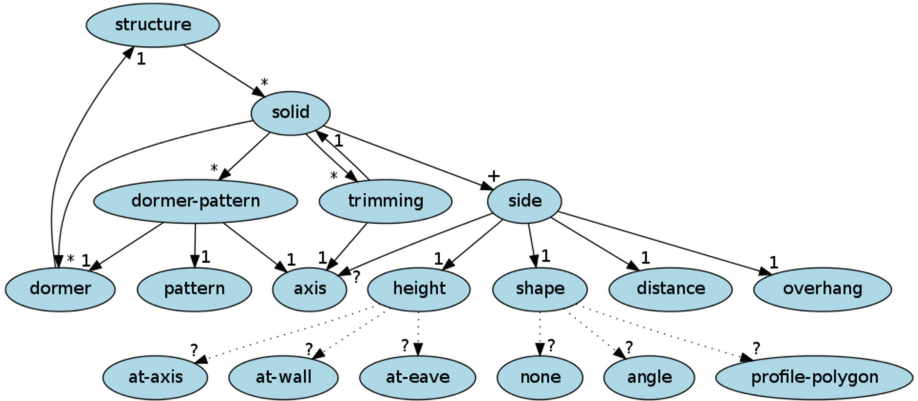
Additionally, a solid can also have dormers (see Fig. 13). A dormer is a structure, which is put on the solid and where appropriate cutting of roof faces is

done on the solid. For repetition of the same dormer multiple times along an axis, we provide the dormer-pattern construct, where a repetition pattern must be provided. These patterns are formed in analogy to repetition or subdivision patterns in shape grammars.

This specification can easily be extended for further building and roof elements (e.g. chimneys, ..) or different side parametrization.



**Fig. 13.** Dormers on a solid. They are structures which are places on a solid, either individually or according to a pattern.



**Fig. 14.** Our abstract building model: The coarse structure is specified by solids and their influence to each other. Nodes of the graph show elements, arrows show their relationships (in a classical *has-a* notation). The quantifiers show the number of sub-elements (? ... zero or one, 1 ... one, \* ... zero or multiple, + ... one or multiple).eps

## 4.2 Geometry Operators

The geometry of roofs must often fulfill certain criteria. Important properties for the roof can be for example:

- Faces are planar (all vertices of a face must be co-planar - this can be a problem when a face has more than three vertices).
- Faces intersect in one common vertex (for more than three faces this can be a problem).
- Ridges are horizontal (often means that the to the ridge adjacent building sides must be exactly parallel).
- Faces merge over multiple building parts (this often means that the corresponding walls must be co-linear).

These properties form parameter dependencies, and they are a problem for the input format, since there must be a way to avoid over-specification of parameters (and therefore violation of the dependencies). Because of that, we included geometry operators in the language. They operate on the axis element and can produce new axis elements that allow for fulfilling the requirements stated above. The operators are for example translation, rotation, parallel-translation, and line-intersection.

## 5 Implementation

### 5.1 Double-Covered Areas

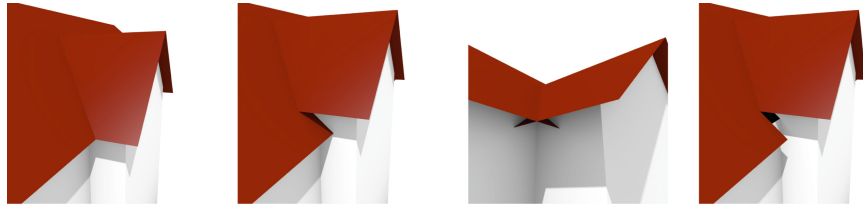
The geometry of a roof can be described as a 2-dimensional manifold where every possible vertical ray intersects the roof exactly 0 or 1 times. This means that the roof geometry can be generated as the surface of solid geometry. The idea is to generate solid geometry (which easily allows Boolean operations) and then extract the roof faces from it. The exception are double-covered areas (see Fig. 15). Here the roof of one part of the house rises over the roof of another part (see Fig. 15(b)), and a vertical ray can intersect the overall roof multiple times, which complicates the algorithm.

At a double-covered area, the lower part of the roof must extend to the wall of the other solid. But when the eaves of both solids are at the same height (see Fig. 15(a)), they are not allowed to extend to the other solids wall, otherwise incorrect geometry on the bottom side of the roof would occur (see Fig. 15(c)). In fact, only the roof part with the lower eave is allowed to extend to the other solids wall.

### 5.2 Organization of the Components

In order to extract the geometry we generate convex polyhedra for each side and perform Boolean operations on them. Figure 16 shows the *inner* and *outer side geometry* for various sides. They are both identical, except that the outer one extends the amount of overhang farther outward from the wall. Through this, we will later be able to generate the geometry for overhangs and double-covered areas.

The geometry for a solid is generated by intersection of the geometry of all sides. Taking either the inner or outer side geometry, this yields the *inner* and



(a) Eaves at same height. No area of the roof is a double covered area. (b) Eaves at different height. A double covered area is formed. (c) Same height - wrong result if the roof geometry extends to the adjacent wall. (d) Different height - wrong result if the roof geometry does not extend to the adjacent wall.

**Fig. 15.** Eaves and double-covered areas. Eaves are a challenge for the geometry generation. Eaves with the same height (a) need a different geometry generation than eaves with different heights (b). We show two examples (c, d) where simple algorithms yield wrong results.

*outer sub-solids* as shown in Fig. 17. The inner sub-solid is used for the extraction of the geometry for the walls, and the outer sub-solid is used for the extraction of the geometry of the roof. An exception are roofs with double-covered areas. Here the roof of one solid is covered by the roof of another solid. So for each side of a solid, depending on the situation, the inner or outer side geometry needs to be taken. Then an individual sub-solid is formed by the intersection of the side geometries.

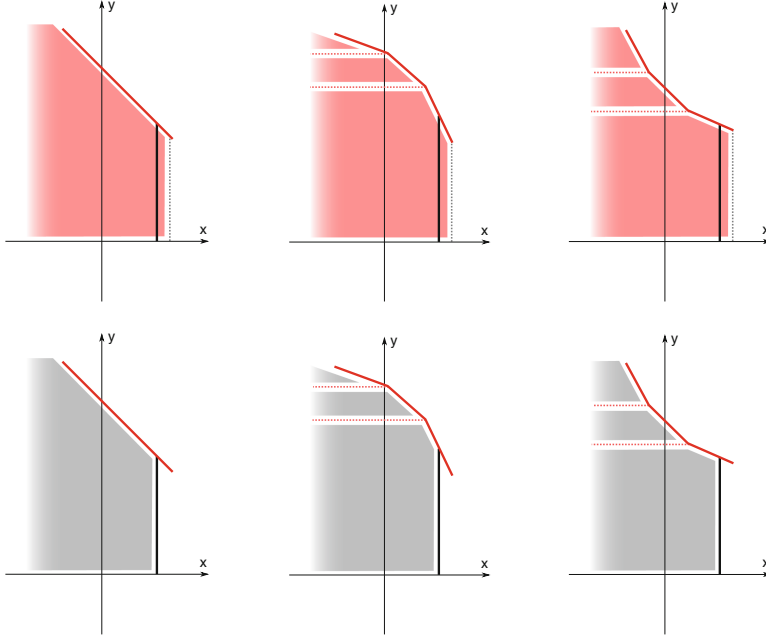
### 5.3 Trimming

As already described, a solid can be trimmed by another solid. Which sides of the other solid are selected for the trimming is described in Sect. 3.4. For the geometry extraction this means that the selected sides of the other solid are simply added as sides to the trimmed solid. This way, the solid gets automatically cut because its geometry is determined by the intersection of the geometry of its sides.

### 5.4 Geometry Extraction

The geometry of the walls can be obtained by performing a Boolean union of the inner sub-solids (see Fig. 17) of all solids, after they are trimmed. Then the sideways facing face of the resulting volume form the walls of the structure.

The geometry of the roof could also be obtained by performing a Boolean union of the outer sub-solids of all solids. Taking all the upward facing faces of the resulting volume would yield the roof geometry. But double-covered areas make the geometry extraction for the roof more complicated. This method would lead to geometry errors (as seen in Fig. 15(d)).



**Fig. 16.** Side geometry. For the geometry generation each side is represented via convex polyhedra (shaded areas). Top: *inner side-geometry*. Bottom: *outer side-geometry*. First column: Simple roof with one roof face per side. Second column: The roof is specified via a convex profile-polygon. Third column: The roof is specified via a concave profile-polygon.

Therefore, for every solid the roof geometry is extracted separately. The algorithm for extraction is shown in Algorithm 1 and the according description is presented in Table 1.

The result of the geometry extraction can be seen in Fig. 18. Note that the roof has the correct geometry, also in the inside of the building (the roofs bottom side).

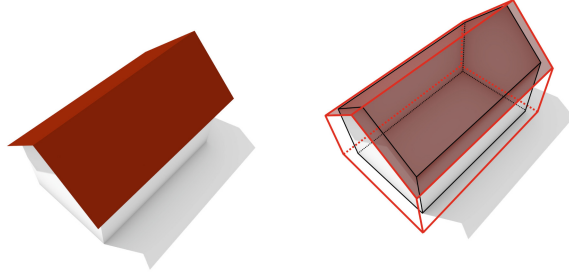
## 5.5 Processing

We have built a system with multiple steps where the input data is transformed. The input format is XML but we built our own short notation that is semantically equivalent but better read- and writable than pure XML.

The steps of our current implementation are as follows:

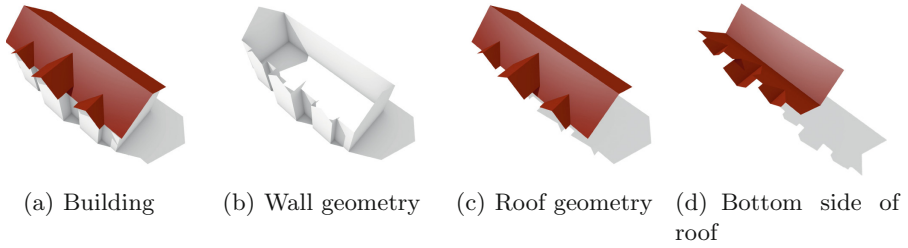
1. The input file in short notation created by the user is converted into XML.
2. The XML file undergoes an enrichment step where sides and parameters are automatically added according to Sect. 3.3.
3. The new XML file is converted into GML (our own in-house programming language for procedural modeling).





**Fig. 17.** Each solid consists of two sub-solids, a wall sub-solid (inner black outline) and a roof sub-solid (outer red outline), which are utilized for the trimming process (Color figure online).

4. Our engine in GML creates the geometry according to the principles described in Sect. 5.



**Fig. 18.** Extracted geometry of a building. Wall and roof geometry are extracted separately. The bottom side of the roof shows that the solids are not simply stuck together, and the geometry is extracted correctly.

## 6 Results and Applications

We evaluated our prototype implementation by modeling three scenes that exhibit interesting roof structures from aerial images. Our workflow consists of the following steps:

1. Trace important lines via a vector-drawing program.
2. Convert lines into axis.
3. Assign parametrized sides with respect to axis.
4. Assign sides to solids.
5. Specify trimmings between solids.

```

1   $S$  = arbitrary structure
2  for  $A \in \text{solids of } S$  do
3      for  $a \in \text{sides of } A$  do
4           $L_{solid}$  = empty list
5          for  $B \in \text{solids of } S \text{ where } A \neq B$  do
6               $L_{side}$  = empty list
7              for  $b \in \text{sides of } B$  do
8                  if  $b$  rises above  $a$  then
9                       $g$  = inner side-geometry of  $b$ 
10                 else
11                      $g$  = outer side-geometry of  $b$ 
12                 end
13                 append  $g$  to  $L_{side}$ 
14             end
15              $I = \cap$  of all in  $L_{side}$ 
16             append  $I$  to  $L_{solid}$ 
17         end
18          $T$  = outer sub-solid of  $A - (\cup \text{ of all in } L_{solid})$ 
19         render the roof face (according to  $a$ ) of  $T$ 
20     end
21 end

```

**Algorithm 1.** Extracting the geometry for the roof of one structure. For simplicity, the algorithm shown here only covers sides with one roof element and not with a profile-polygon (as in Fig. 12).

The result is an abstract building description in an XML file that reflects our specification (see Fig. 14). As of now, the workflow includes some manual steps, for example, the assignment of parametrized sides. We observed that while the models have complex roof cases, the roof shapes follow simple geometric rules when blended together. As it was expected, the decomposition into convex parts turned out to be quite obvious. Evaluation times were measured on the CPU implementation of our prototype system on a 2.6 GHz quad core machine.

The first model is the royal palace of Milan (Fig. 19) where we also placed some dormers on the roof. The model consists of 28 solids (excluding dormers), and its evaluation took 6 s.

The second model is a part of the city of Graz, Austria (Fig. 20). It consist of 7 structures that have 34 solids in total, and its evaluation took 5 s. The roofs have considerable irregularities, which became a challenge in the modeling process.

The third model is the Magdalena palacio in Santander (Figs. 1 and 21), which is assembled from 25 solids, and its evaluation took 8 s without the facade detail and 20 s with the facade detail. The roof of this building exhibits some interesting features, such as the cavities on the middle part of the building, which are hard to model using fully automatic (e.g., straight skeleton based) approaches.

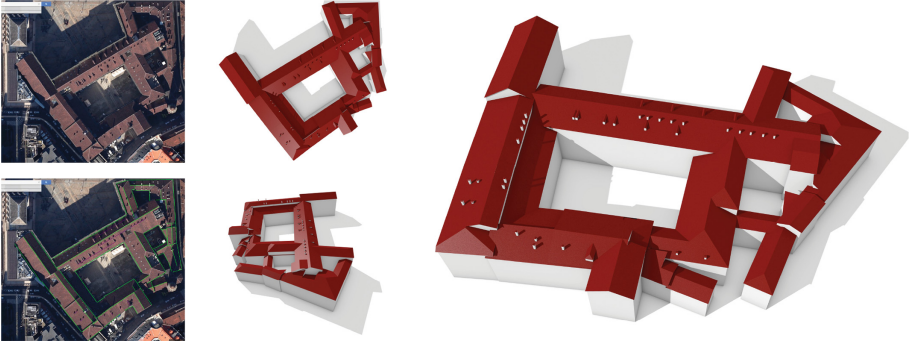
**Table 1.** Description of Algorithm 1.

Line	Description
1	In our system different structures have independent roofs so we can calculate the roof independently for one structure
2,3	We calculate the roof face for each side of a solid separately
4	The roof of the solid changes when another solid intersects with it. We will trim the solid with all other solids, but because of possible double-covered areas we have to take either the inner or outer side-geometry of another solid, depending on the situation. Therefore we create a new list where we will save the new other solids with inner/outer side-geometry
5	We iterate over all other solids of the house
6–13	For every side of the other solid we decide whether it rises over the side we are currently calculating (and therefore forms a double-covered area). The test for rising over the other side is described in Sect. 5.1. If yes, we take the inner side-geometry. If no, we take the outer side-geometry. We collect the side-geometries in a new list
15–16	We build the Boolean intersection of all the elements in the list and thereby form a custom sub-solid that respects the double-covered areas for each of its sides
18	All the custom sub-solids are now Boolean subtracted from the outer sub-solid of our initial solid. This removes all areas from the roof where another solid intersects
19	The new roof of this side can now be rendered

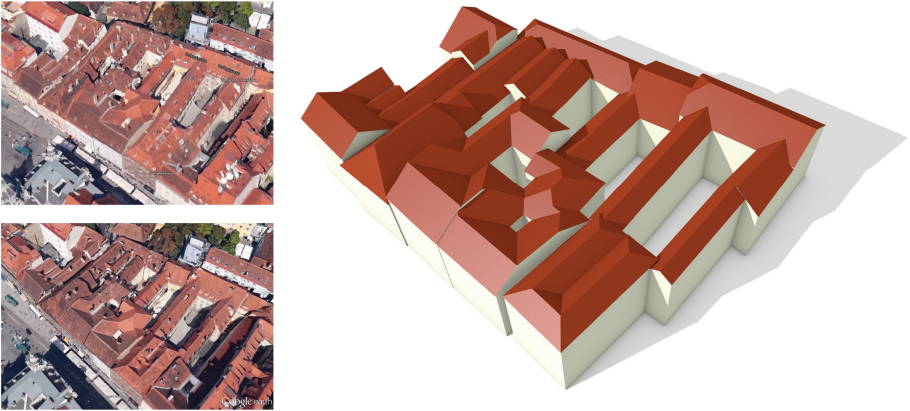
When doing direct comparison, the third model exhibits a more constrained structure: the ridges follow two principle axes which are perpendicular to each other. So we realigned the traced lines from the aerial view with the operators described in Sect. 4.2. A more irregular layout, like that in the first and second model, can make the modeling harder; at regions where solids adjoin or intersect, artifacts are more likely to arise.

For all models, we did not have any measurements, therefore we estimated values for missing parameters, e.g., the side heights, by using photographs of the buildings as guidelines. While in this case the result is not a fully accurate model, it produces a good approximation and visually satisfying result (e.g., see Fig. 21). Due to the missing parameters, we had to try different parameter sets several times to ensure that the roof has the right topology. Nevertheless, our abstract definition allows for quick reparametrization if real measurements become available.

The geometry can be exported to .obj or other file formats for further usage in other programs. We did our renderings in Blender where materials and lighting were defined.



**Fig. 19.** Model of the royal palace of Milan. For this model, the modeling process was done in four steps: (1) obtaining an aerial image of the building (top left); (2) manual tracing of important lines in a vector-drawing program (bottom left); (3) extraction of the vector coordinates to the XML format; (4) manual augmentation of remaining parameters in the XML format (like height, roof angle, etc.). The resulting model was created by our prototype implementation, and then rendered (top middle, bottom middle, right). Aerial images: Imagery ©2015 Google, Map data ©2015 Google.



**Fig. 20.** The inner city of Graz, Austria. The detail is the same as in Fig. 4. Top left: 3D view from Google maps. Bottom left: 3D view from Google earth. Right: Our model. While the models from Google have high detail, they do not ensure constraints like planarity of roof faces or perfectly straight ridges, eaves, or other edges. Our model ensures this by construction. The modeling process was done with aerial images as in Fig. 19. Aerial images: Imagery ©2015 Google, Map data ©2015 Google.



**Fig. 21.** Model of the Magdalena palace in Santander, Spain (as in the teaser - Fig. 1). The modeling process was done with aerial images (middle left, middle middle) as in Fig. 19. The detail of the model is increased with existing procedural approaches. Here we used a conventional shape grammar approach for modeling the facades (top). The model has three cases where roof topology changes because of a specified trimming (middle right - marked with yellow circles). (a) is the top row case from Fig. 9, (b) is the middle row case from Fig. 9, and (c) is a complex case where five solids with different trimming specifications are involved. The bottom row shows the 3D model in Google maps on the left. Again the constraints of planarity of roof faces and straightness of edges are not satisfied. Two additional regions with wrong geometry are emphasized in yellow circles. Aerial images: Map data ©2015 Google, basado en BCN IGN España.

## 7 Conclusion

Modeling the coarse structure of a building by the composition of parts proved to be suitable for buildings of classical architectural style. However, in many cases a building is not a simple union of such parts. In this paper, we proposed automatic trimming of adjacent parts, which facilitates a concise, abstract decomposition of a building into parts and their geometric influence. A coarse 3D model of a building can then be generated from such a description, which can further be fed into a conventional pipeline (e.g. shape grammars) that generates detailed geometry for building parts (e.g. windows and doors), see Fig. 21. Our abstract building description facilitates easy re-parametrization of building parts, as the 3D model just has to be re-evaluated after a parameter has been changed (as seen in Fig. 11).

We see applications for our method in production pipelines of virtual worlds. This includes the digitalization of cultural heritage, since it can reproduce the complex geometry of roof landscapes in historic cities. It is also suitable for the movie and video games industry, where sometimes man years of work have to be spent on modeling the environment. Our approach, which integrates into existing procedural methods, further enhances the expressiveness of those methods and advances the automatic generation of geometry.

## 8 Future Work

The main focus in future work is directed towards more automation in the modeling process. One direction is the automatic generation of solid and side descriptions by a rule based system (e.g. stochastic grammars). The other direction is headed towards automatic reconstruction by automatic tracing of important roof lines and the incorporation of additional measurement data for automatic parametrization of building sides (e.g., height of solids). Another interesting direction for future research poses the incorporation of more complex geometry (curves) into the cross-sectional profiles of building sides.

## References

1. Aichholzer, O., Aurenhammer, F., Alberts, D., Gärtner, B.: A novel type of skeleton for polygons. *J. Univ. Comput. Sci.* **1**, 752–761 (1996). Springer
2. Ameri, B., Fritsch, D.: Automatic 3d building reconstruction using plane-roof structures. In: ASPRS, Washington, D.C. (2000)
3. Aurenhammer, F.: Weighted skeletons and fixed-share decomposition. *Comput. Geom.* **40**(2), 93–101 (2008)
4. Dörschlag, D., Gröger, G., Plümer, L.: Semantically enhanced prototypes for building reconstruction. In: Stilla, U., et al. (eds.). *Proceedings of 36th Photogrammetric Image Analysis, PIA 2007. International Archives of ISPRS* (2007)
5. Finkenzeller, D.: Detailed building facades. *IEEE Comput. Graph. Appl.* **28**(3), 58–66 (2008)



6. Finkenzeller, D.: Modellierung komplexer Gebäudefassaden in der Computergraphik. KIT Scientific Publishing (2008)
7. Fischer, A., Kolbe, T.H., Lang, F., Cremers, A.B., Förstner, W., Plümer, L., Steinhage, V.: Extracting buildings from aerial images using hierarchical aggregation in 2d and 3d. *Comput. Vis. Image Underst.* **72**(2), 185–203 (1998)
8. Grün, A.: Semi-automated approaches to site recording and modeling. *Int. Arch. Photogram. Remote Sens.* **33**(B5/1; PART 5), 309–318 (2000)
9. Huang, H., Brenner, C.: Rule-based roof plane detection and segmentation from laser point clouds. In: Joint Urban Remote Sensing Event (JURSE 2011), pp. 293–296. IEEE (2011)
10. Huang, H., Brenner, C., Sester, M.: 3d building roof reconstruction from point clouds via generative models. In: Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, pp. 16–24. ACM (2011)
11. Huang, H., Brenner, C., Sester, M.: A generative statistical approach to automatic 3d building roof reconstruction from laser scanning data. *ISPRS J. Photogram. Remote Sens.* **79**, 29–43 (2013)
12. Kelly, T., Wonka, P.: Interactive architectural modeling with procedural extrusions. *ACM Trans. Graph. (TOG)* **30**(2), 14 (2011)
13. Kim, K., Shan, J.: Building roof modeling from airborne laser scanning data based on level set approach. *ISPRS J. Photogram. Remote Sens.* **66**(4), 484–497 (2011)
14. Krecklau, L., Kobbelt, L.: Procedural modeling of interconnected structures. *Comput. Graph. Forum* **30**, 335–344 (2011). Wiley Online Library
15. Krecklau, L., Pavic, D., Kobbelt, L.: Generalized use of non-terminal symbols for procedural modeling. *Comput. Graph. Forum* **29**, 2291–2303 (2010). Wiley Online Library
16. Laycock, R.G., Day, A.: Automatically generating roof models from building footprints (2003). <https://otik.zcu.cz/handle/11025/991>
17. Liu, Y., Xu, C., Pan, Z., Pan, Y.: Semantic modeling for ancient architecture of digital heritage. *Comput. Graph.* **30**(5), 800–814 (2006)
18. Merrell, P., Schkufza, E., Koltun, V.: Computer-generated residential building layouts. *ACM Trans. Graph. (TOG)* **29**(6), 181 (2010)
19. Milde, J., Brenner, C.: Graph-based modeling of building roofs. In: AGILE Conference on GIScience (2009)
20. Milde, J., Zhang, Y., Brenner, C., Plümer, L., Sester, M.: Building reconstruction using a structural description based on a formal grammar. *Int. Arch. Photogram. Remote Sens. Spatial Inf. Sci.* **37**, 227–232 (2008)
21. Müller, P., Wonka, P., Haegler, S., Ulmer, A., Van Gool, L.: Procedural modeling of buildings. *ACM Trans. Graph.* **25**, 614–623 (2006). ACM
22. Stiny, G., Gips, J.: Shape grammars and the generative specification of painting and sculpture. In: IFIP Congress, vol. 2, pp. 1460–1465 (1971)
23. Sugihara, K., Hayashi, Y.: Automatic generation of 3d building models from building polygons on GIS. In: ICCCBEXI Proceedings of the 11th ICCCBEXI, Montreal, Canada, pp. 14–16 (2006)
24. Sugihara, K., Hayashi, Y.: Automatic generation of 3d building models with multiple roofs. *Tsinghua Sci. Technol.* **13**, 368–374 (2008)
25. Taillandier, F.: Automatic building reconstruction from cadastral maps and aerial images. *Int. Arch. Photogram. Remote Sens.* **36**(3/W24), 105–110 (2005)
26. Teoh, S.T.: Generalized descriptions for the procedural modeling of ancient east asian buildings (2009). <http://dl.acm.org/citation.cfm?id=2381290>

27. Thaller, W., Krispel, U., Zmugg, R., Havemann, S., Fellner, D.W.: Shape grammars on convex polyhedra. *Comput. Graph.* **37**(6), 707–717 (2013)
28. Verma, V., Kumar, R., Hsu, S.: 3d building detection and modeling from aerial lidar data. In: *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2213–2220. IEEE (2006)
29. Wonka, P., Wimmer, M., Sillion, F., Ribarsky, W.: Instant architecture. *ACM Trans. Graph.* **22**(3), 669–677 (2003). doi:[10.1145/882262.882324](https://doi.org/10.1145/882262.882324). <http://doi.acm.org/10.1145/882262.882324>



Transactions on Computational Science XXVI

Special Issue on Cyberworlds and Cybersecurity

Gavrilova, M.L.; Tan, C.J.K.; Iglesias, A.; Shinya, M.;

Galvez, A.; Sourin, A. (Eds.)

2016, XV, 173 p. 90 illus. in color., Softcover

ISBN: 978-3-662-49246-8