MODELLING 3D OBJECTS USING WEAK CSG PRIMITIVES

Claus Brenner

Institute of Cartography and Geoinformatics, University of Hannover, Germany – claus.brenner@ikg.uni-hannover.de

KEY WORDS: LIDAR, Urban, Extraction, Modeling, Reconstruction, City, Semi-automation.

ABSTRACT

There have been numerous approaches to automatically extract man-made objects from images or laser scan data in the past. Many efforts have been put into the detection and measurement of object parts. However, putting those parts together in such a way that a topologically correct object representation is obtained – which additionally meets certain regularization conditions – has not been addressed in a satisfactory manner. In this paper, following a review of existing methods for obtaining roof topologies of buildings, an approach termed 'weak constructive solid geometry (CSG) primitives' is proposed. This approach is intended to fill the gap between B-rep and CSG modelling, combining advantages of both.

1 INTRODUCTION

When research on man-made structures approached the problem of automatically deriving building models from sensor data about 20 years ago, the use of aerial images to obtain roof structures was common. In the meantime, new sensors have become operational, such as aerial and terrestrial laser scanners, extending the possibilities to acquire data. However, the requirements have increased also, nowadays for example there is a demand for including textured and geometrically detailed façades.

1.1 Object Representation

What is a 'good' virtual representation of a real world object? For example, given a dense point cloud from laser scanning or image matching, one might argue that the best boundary representation would be a triangulated surface consisting of those points, or a subset thereof. This approach is typically used for free-form objects, where the original measured point cloud is triangulated, smoothed, reduced, etc..

Usually, this is not the description one strives for in the context of modelling man-made objects such as buildings, for various reasons:

- Even when used only for visualization, densely triangulated meshes are not well suited in the context of buildings, since the human eye is quite sensitive to deviations between the (often piecewise flat) object and its approximation by the densely triangulated surface. Also, standard triangle reduction algorithms are usually not well suited for polyhedrons. A better visual impression can be obtained by adding texture (e.g. (Früh and Zakhor, 2003)), however, the model might still look 'jagged', especially at building borders.
- For graphics performance, storage, and transmission, a surface description consisting of as few polygons as possible is desirable.
- Since 3D building models will become a part of 3D geographic data bases, an object-wise representation is more adequate than a huge triangulated surface. As with 2D databases nowadays, different scales will be available, which correspond to different levels of detail (LOD). It would be desirable to model buildings in such a way that different LOD's can be derived automatically.

As a result, the challenge is to obtain an object representation from dense measurements, which represents the major structure of an object, given a certain desired generalization level. In contrast to the reduction of point clouds or triangulated surfaces, this involves an interpretation step.

1.2 B-rep vs. CSG

A large percentage of buildings can be represented by polyhedrons. Two main approaches to model such buildings have evolved in the past. The first constructs the boundary from measured points, lines or planes, thus builds the boundary representation (B-rep) directly. The second obtains the object by combining volume primitives using Boolean operations (constructive solid geometry, CSG). From the CSG representation, the B-rep can be obtained automatically and unambiguously, when required.

CSG modelling is used widely in computer aided design (CAD), since it has a number of advantages: *(i)* modelling using primitives and Boolean operations is much more intuitive than specifying B-rep surfaces directly (except for situations where the set of primitives is not appropriate for a given object), *(ii)* the primitives can be parameterized, thus enabling reuse and collection in libraries, *(iii)* they can be associated with other, additional information, and *(iv)* the CSG modelling tree contains implicit information that can be used for many purposes.

When building CSG models on the basis of measured data, an additional advantage is that due to the implicit geometrical constraints of the primitives, the determination of primitive parameters is quite robust. However, one has to keep

in mind that since a conversion from a B-rep to a CSG representation is ambiguous, deriving a CSG representation from given (measured) data is ambiguous, too – unless additional hints can resolve this ambiguity.

Looking at different modelling approaches, the following points are of importance:

1. How is topological correctness of the surface ensured?
2. How are geometric constraints such as meeting surfaces, roofs of same slope, parallelism and rectangularity enforced?
3. How is the desired level of generalization obtained, i.e. which measures are taken to distinguish between 'important' parts that should be modelled and 'unimportant' parts, which have to be left away.

When object representations are built using CSG, this solves a number of issues. First, primitives themselves are assumed to be topologically correct and Boolean operations will preserve correctness. Second, geometric constraints hold for the primitives implicitly: for example, the faces of a box are parallel or normal to each other. However, this does not extend by default to aggregates of primitives. Third, by imposing limits on the allowed primitive sizes, the generalization level can be controlled.

2 PREVIOUS WORK

2.1 Example Building Reconstruction Systems

This section reviews a selected number of building reconstruction approaches with special focus on how they deal with topological correctness, geometric constraints, and generalization. Table 1 summarizes the main points.

(Haala and Brenner, 1997) derive a skeleton (Aichholzer et al., 1995) from a given ground plan, which defines regions of interest in the digital surface model (DSM). Analyzing these regions, roof surfaces are accepted or rejected, and the skeleton is rebuild using only accepted surfaces. The topological correctness is ensured by the skeleton algorithm, which is constructive with well-defined result. Geometric constraints are not imposed, which can lead to wrong gable positions, especially when the ground plans are displaced relative to the DSM. Since each ground plan segment leads to at most one roof surface patch, the generalization level is implicitly determined by the ground plan.

(Weidner, 1997) extracts roof faces using a DSM segmentation. No topology is build. However, it is proposed to automatically derive mutual relationships between the extracted faces, such as 'same slope', 'symmetry', and 'antisymmetry', and to insert them as constraints into a global robust adjustment. There is no control for the generalization level except for the minimum accepted roof face size.

(Grün and Wang, 2001) extend a previous approach described in (Grün and Wang, 1998). It is based on conventional, manual measurement of corresponding image points in stereo images. For all vertices of the object to be acquired, corresponding points have to be measured. The resulting point cloud is termed "weakly structured", since the operator can give hints regarding the roof topology by measuring in a certain order and placing the points in different layers. Then, an automatic process based on relaxation recovers the roof topology from the point cloud. Finally, constraints are formulated, e.g. all eaves points having the same height, vertices of a face lying in a plane, etc.. A least squares adjustment is used to estimate final point locations. It seems that no constraints are used to enforce relationships between different roof parts, as overlapping parts are corrected by a snap operation which is done after the adjustment. Altogether, the approach builds the surface of the object directly by a mixed manual and automatic procedure. Regularity is enforced by some constraints as well as a snap. The generalization level is entirely controlled by the operator.

(Gülch and Müller, 2001, Gülch et al., 1999) describe an approach which is based on CSG primitives measured semi-automatically in aerial images. Buildings are modelled using a fixed number of parametric primitives like flat-, desk-, saddleback-, hipped-roof etc., which are combined by Boolean operations. The selection of each appropriate primitive is carried out manually by an operator. Then, the wireframe model is overlaid in two images and the operator can adapt the parameters accordingly. The adaption is supported by various image matching tools so that only a few mouse clicks are necessary to instantiate and measure a primitive. Topological correctness is enforced by the CSG approach. Regularity is enforced implicitly by the primitives. Across primitives, a snapping procedure is used. Generalization is controlled entirely by the operator.

(Vosselman, 1999) extracts faces from non-regularized laser scan data using a Hough transform, followed by connected component analysis. From this, edges are found by intersection of faces and analysis of height discontinuities. No ground plans are used as additional information, however since jump edges cannot be determined very well from laser scanning, a 'main building orientation' is derived and used as a constraint for the edge orientation. The topology is build by bridging gaps in the detected edges. The use of geometric constraints is proposed. Generalization is again controlled by a minimum face size.

(Brenner, 1999) divides ground plans into 2D primitives (in this case, rectangles) using a heuristic algorithm. For each of the 2D primitives, an optimal 3D primitive is selected from a fixed set of available roof types, and its parameters are estimated using the DSM. The primitive selection and parameters can be changed later on using a semiautomatic extension. The final object representation is obtained by merging all 3D primitives. Roof topology is generated by the Boolean merge operation. Adjacent 3D primitives with similar eaves heights are 'snapped' to exactly the same heights, however no constraints are formulated and no combined adjustment takes place. The initial 2D primitive generation can be controlled by a buffer parameter which will omit primitives for small ground plan

	Topology	Regularization	Generalization	Description
1	Constructed: skeleton of ground plan.	—	No. of roof faces bounded by ground plan edges.	Regular DSM, ground plans. Hypothesize-and-test using skeleton. Result strongly coupled to ground plans.
2	—	Constraints derived automatically.	Controlled by minimum region size.	Regular DSM. Segmentation of planes but no topology built. Automatic derivation of constraints.
3	Relaxation to connect points from weakly structured cloud.	Constraints, snap.	Manually by operator.	Stereo images. Manual measurement of weakly structured point cloud, relaxation to derive topology, adjustment using constraints, snapping to correct topology. Semiautomatic.
4	CSG: Primitives and Boolean operations.	CSG primitives, snap.	Manually by operator.	Mono images. Selection of primitives by operator, measurement of primitive parameters supported by image matching. Semiautomatic.
5	Find and connect edges between extracted planar faces.	Outlines and jump edges follow main orientation. Constraints proposed.	Controlled by minimum region size.	Original laser scan data. Hough based region extraction, detection of edges, connected edges form topology.
6	CSG: Primitives and Boolean operations.	CSG primitives, snap (limited to height).	Influenced by ground plan and buffer parameter.	Regular DSM, ground plans. Subdivision of ground plan into rectangles, reconstruct individually, and merge.
7	Constrained tree search to find topology between extracted planar faces.	Constraints proposed.	Influenced by ground plan and acceptance rules.	Regular DSM, ground plans. Extract planes, accept/reject on the basis of rules, global search for topology. Weakly coupled to ground plans.
8	Find and connect edges between extracted planar faces.	—	Influenced by ground plan and split & merge parameters.	Original laser scan data. Subdivide building area according to ground plan, extract faces using Hough transform, split & merge. Detection and connection of edges.
9	Find and connect edges between extracted planar faces.	Automatic constraint detection and global adjustment proposed.	Controlled by minimum region size.	Regular DSM for segmentation, original laser scan points for estimation. Extraction of roof planes, merge, detection and connection of edges.

Table 1: Comparison of different modelling approaches: How is the topology obtained, how are regularities enforced? 1 (Haala and Brenner, 1997), 2 (Weidner, 1997), 3 (Grün and Wang, 1998), 4 (Gülch et al., 1999), 5 (Vosselman, 1999), 6 (Brenner, 1999), 7 (Brenner, 2000a), 8 (Vosselman and Dijkman, 2001), 9 (Rottensteiner and Briese, 2003).

extrusions. Thus, the generalization level can be controlled, but is of course tied closely to the ground plan.

(Brenner, 2000b) extracts planar faces from a regularized DSM using a random sampling consensus (RANSAC) approach. Faces are accepted or rejected based on a set of rules, which express relationships between faces and ground plan edges. The final topology of the roof is obtained from all accepted regions by a global search procedure. The introduction of constraints and a least squares adjustment to enforce regularity is described in (Brenner, 2000a). Generalization is linked to the ground plan and the set of rules.

(Vosselman and Dijkman, 2001) and (Vosselman and Suveg, 2001) is an approach similar to (Vosselman, 1999), however to prevent spurious roof faces, ground plans are introduced as additional information. Concave ground plan corners are extended to cut the building area into smaller regions. The Hough-based plane extraction is constrained to those regions. Split-and-merge is used to obtain the final faces. By using ground plans, generalization is tied to the ground plan generalization, but also depends on the parameters during split-and-merge.

(Rottensteiner and Briese, 2003) extract roof races using seed regions and region growing in a regularized DSM. Similar to (Vosselman, 1999), intersection and step edges are detected and a polyhedral model is derived. It is proposed to detect regular structures automatically and to introduce them as constraints into a global adjustment. Generalization is controlled by parameters governing the plane extraction process.

2.2 Conclusions Drawn

From the presented approaches, one can conclude that building the correct topology, enforcing geometric regularities and ensuring a given generalization level are major problems that have not been solved yet in a satisfactory manner.

The easiest way to ensure a correct surface *topology* is to construct the boundary representation directly. However, this is only true as long as no subsequent processes (snapping, parameter estimation) lead to a geometric change which affects topology. Also, adjacent buildings should not be modelled individually. Constructive algorithms like CSG Boolean operations or building the skeleton yield the correct topology, provided no numerical instabilities arise (de Berg et al., 2000). When using CSG, the primitive parts from which an object is built must be aligned properly, which is often a problem when the parameters of the primitives are determined from measurements.

Enforcing *constraints* has been proposed by several authors, however it has not been used to a larger extend. The practical problem with constraints is that their number increases quickly with scene complexity. For example, similar to the 2D case outlined below, a simple box in 3D space can be described by its position (3), orientation (3) and dimensions (3), for a total of 9 parameters, or degrees of freedom (DOF). However, considering this box as a general polyhedral surface, we obtain 24 DOF for the 8 points, 18 DOF for the 6 planes, and 33 constraints which enforce regularity, so that again 9 DOF remain. Thus, even if one had a modeler capable of identi-

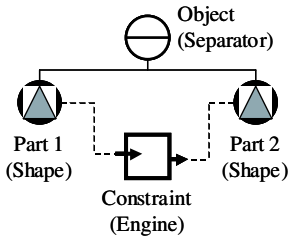

Figure 1: Simple scene graph of an object consisting of two shapes and an engine which connects their fields. The drawing uses scene graph symbols from Open Inventor.

fying constraints from reconstructed object parts automatically, it would probably be prohibitive to check interactively if the correct constraints have been identified, and, given cross-dependencies, which constraints are missing.

Finally, *generalization* is a mostly open problem. Especially approaches based on a general planar segmentation usually use little more than region sizes to determine if a region should become part of the final object. Some approaches use ground plans which influence generalization, (Brenner, 2000b) uses a set of rules to describe simple roofs. On the other hand, CSG modelling has the advantage that primitives are tied to interpretations, which facilitates the definition of "valid" roofs.

3 SCENE GRAPHS

Scene graphs have been used extensively in computer graphics. Their primary purpose is to describe a scene for rendering (Wernecke, 1994, OpenSceneGraph, 2004). A scene graph is a tree structure, able to store a hierarchical description of a scene in terms of nodes. Objects are modelled in parts, which are usually parameterized and can be reused across the graph.

For the purpose of reconstructing objects from measurement data, three scene graph concepts are of particular interest: fields, engines and node kits. *Fields* are used to store parameters of nodes. Since nodes are actually classes, parameters could also be stored in simple class member variables. However, defining fields has the advantage that *i)* parameters of all nodes can be accessed in a unified way, *ii)* field changes can be monitored by the system, and *iii)* fields can be connected to each other (Wernecke, 1994). *Engines* provide this possibility to connect node fields. In scene graphs, one of their main purposes is to animate scenes. However, engines can do arbitrary computations and thus influence a number of result fields based on a number of input fields. Figure 1 shows a simple graph involving an engine. Finally, *node kits* are nodes grouped in a subgraph which can be inserted into the scene graph as one. Node kits somehow resemble libraries of object parts as used in CSG modelling. However, their flavor is a little bit different: a node kit contains a *catalog* of nodes, arranged in proper order, of which only a minimal subset is used by default. Additional nodes in the kit are "switched on" as soon as their field values are set.

4 WEAK PRIMITIVES

4.1 The Idea Behind Weak Primitives

From the conclusions drawn in section 2.2, CSG modelling has a number of advantages: correct surface topology is enforced by the primitives themselves, one has not to deal with a confusing number of constraints because they are implicit, and a handling of generalization is more straightforward since CSG primitives are associated with an interpretation. The resulting CSG tree is certainly a more valuable outcome than a collection of boundary patches, e.g. when different levels of detail of an object are to be derived afterwards. Furthermore, CSG modelling is less sensitive to errors in measurement data and even to data gaps, due to the strong implicit constraints.

The drawback is that those strong constraints cannot be "relaxed" when needed, since they are a inherent property of the primitives. However, real-world man-made objects often have deviations from the idealized shape, which are cumbersome to model using ideal primitives.

The second drawback is that the different CSG primitives which make up an object have to be aligned properly. When primitive parameters are estimated by a fit to real data, this alignment is usually not fulfilled. Performing a "snap" operation afterwards will align properly, but will loose the best fit property. Thus, a single process is desirable, which enforces alignment and a best fit simultaneously.

The idea of weak primitives is to combine the best of CSG and "faces and constraints". Weak primitives are a set of unknowns and a set of constraints, packaged as a single "primitive". Using a weak primitive is just as using a "real" primitive, although internally, the constraints are enforced by equations rather than by an implicit model. The difference becomes obvious when constraints are to be relaxed: weak primitives allow to "switch off" constraints and thus are able do deviate from their ideal shape.

In order to enforce regularities between different primitives, additional constraints have to be imposed. To achieve this, weak primitives expose an interface consisting of fields. Fields of different primitives can be connected by constraints.

One can see the similarity to scene graph concepts outlined in section 3: Node fields correspond with weak primitive fields, engines connecting fields correspond with constraints connecting fields, and weak primitives are similar to node kits in the sense that they contain a number of "prewired" objects which can be turned on or off.

4.2 An Example in 2D

A rectangle in 2D will serve as an example to illustrate the approach. A "CSG representation" of a rectangle involves the parameters (x_c, y_c) for the position, an angle a for the rotation and the width w and height h of the rectangle, for

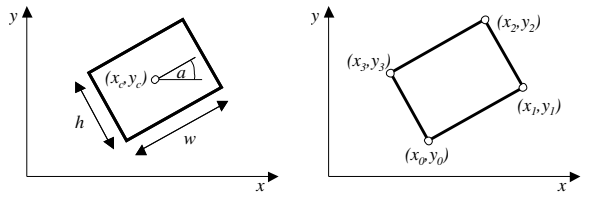

Figure 2: Description of a rectangle in 2D. Left: using 5 parameters. Right: using 4 corner points.

a total of 5 parameters (figure 2, left). From those 5 parameters, all vertices and edges of the rectangle can be derived.

The rectangle can also be described by its vertices (points), edges (lines), and constraints (figure 2, right). If the vertices are given, the edges are defined as well and vice versa. Using vertices as unknowns, one obtains the 8 unknowns (x_i, y_i), $1 \leq i \leq 4$. Since two of the edges are perpendicular and one is parallel to the first edge, 3 constraints apply, so that again 5 parameters remain. Formulating constraints can be done in different ways. Especially in projective geometry, linear constraints can often be obtained (Heuel, 2002). However, in this case, the points as well as the lines are unknown which leads to bilinear equations, as used e.g. by (Brenner, 2000b) or (Grün and Wang, 2001) in the context of building models. The constraint equations thus are linearized and have to be iterated. Four line equations given in Hesse normal form (HNF) $a_i x + b_i y + c_i = 0$, $1 \leq i \leq 4$ yield 12 additional parameters and 4 additional constraints for normalizing the normal vectors, $a_i^2 + b_i^2 = 1$. Two points are on each line, leading to two equations of the form $a_i x_j + b_i y_j + c_i = 0$. Three other equations hold for the normal vectors of the lines, enforcing parallel and perpendicular edges. Table 2 summarizes unknowns and constraints for this case.

#	Description	U	C
4	Points: (x_i, y_i)	8	
4	Lines: $a_i x + b_i y + c_i = 0$	12	
4	Normal vector length = 1: $a_i^2 + b_i^2 = 1$		4
8	Point on line: $a_i x_j + b_i y_j + c_i = 0$		8
2	Perpendicular normals: $a_i a_j + b_i b_j = 0$		2
1	Parallel normal: $a_0 b_2 - b_0 a_2 = 0$		1
Total unknowns		5	

Table 2: Unknowns (U) and constraints (C) for a simple rectangle in 2D.

4.3 Packaging Constraints as Weak Primitives

If one "packages" unknowns and constraints as weak primitives, some of the constraints will become invisible: enforcing the normal vector length and points lying on their respective lines are internal constraints which a user won't change. Parallel and perpendicular constraints will become properties of the object to be switched on and off. The point coordinates, being unknowns, will become fields which can be connected to fields of other primitives. What about the HNF line coefficients? In the above formulation, they are needed when regularization conditions should be met. However, after performing an adjustment, their estimated values are not used, since the object is defined by its points only, which are estimated as well. Thus, a line has

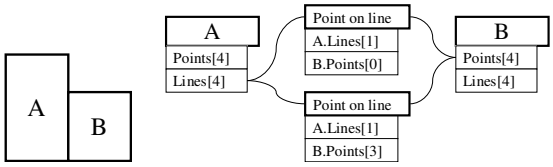

Figure 3: Two rectangles, A and B, standing side by side. Left: geometry. Right: user view of constraints between A and B.

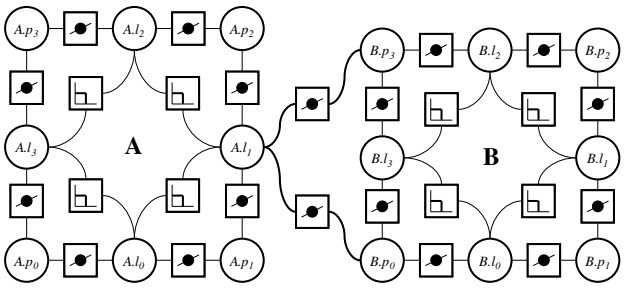

Figure 4: Internal representation of the two rectangles of figure 3 as a graph connecting unknowns by constraints. Unknowns are depicted as circles, constraints as boxes. The icons represent the "rectangular" and "point on line" constraints.

only to take part in the estimation if *i)* regularization conditions are used which involve the line, or *ii)* the line is a field which is connected by a constraint to another primitive.

Thus, one sees that fields offered by a primitive are independent from the parameters which represent its geometry. If a field directly corresponds to a parameter, this *parameter* will be introduced as unknown into the estimation. If not, the *field* will be introduced as unknown and equations will be added relating the unknowns to the parameters of the primitive. In fact, both the "5-parameter" and the "4-point" representations can define the same interface in terms of the fields which are available for connection by constraints.

Figure 3, left shows an example. Two rectangles are required to stand side by side. In order to make them precisely aligned, constraints are introduced, say in this case that the two left points of rectangle B have to lie on the right line of rectangle A. So from an operator's point of view, the rather simple figure 3, right, reflects the scene structure: two objects, A and B, are present which are connected by constraints of type "Point on line".

Internally, however, if the rectangles are represented by four points, the structure of unknowns and constraints more complex. Figure 4 depicts the graph of unknowns and constraints which results. Each time an operator interaction changes parts of the graph, the required graph nodes are determined, unknowns and constraints are set up, and an estimation is performed. Figure 5 shows how the two example primitives react to different user interactions.

5 CONCLUSIONS AND OUTLOOK

In this paper, a new modelling approach termed weak CSG modelling is proposed. Its main idea is to package, or hide,

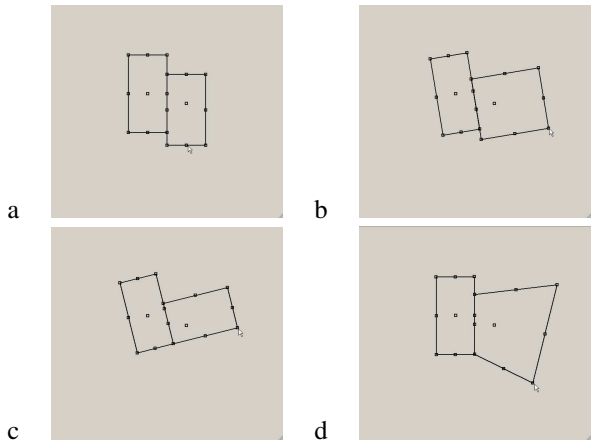

Figure 5: Constraints at work: while the user modifies the scene, constraints are simultaneously set up and solved. (a) Scene with constraints, as defined in figure 3. (b) When the primitives are turned, they keep aligned. (c) An additional constraint is inserted which links the lower right point of A to the lower left point of B. (d) The "rectangular" constraints are switched off for B, but the point and line stay aligned.

regularity constraints of objects inside black boxes which appear to the user as traditional CSG primitives. Thus, the advantages of CSG and B-rep modelling are combined.

There is much room for extensions to obtain a truly interactive system. Even though there are not many constraints to be "wired" by a user, inserting them should be very intuitive – and not be associated with editing a graph representation of the scene. For example, when a user drags an object part close to another object, the system should automatically propose the constraints which make sense between the two objects involved. The same holds for associating primitives with measured or pre-segmented data. The next step will be to apply the approach to a truly 2D/3D setup, where image and laser scan data from aerial and terrestrial sensors are combined.

ACKNOWLEDGEMENTS

This work has been funded by the VolkswagenStiftung, Germany, http://www.volkswagenstiftung.de/.

REFERENCES

Aichholzer, O., Aurenhammer, F., Alberts, D. and Gärtner, B., 1995. A novel type of skeleton for polygons. Journal of Universal Computer Science 1(12), pp. 752–761.

Brenner, C., 1999. Interactive modelling tools for 3D building reconstruction. In: D. Fritsch and R. Spiller (eds), Photogrammetric Week 99, Wichmann Verlag, pp. 23–34.

Brenner, C., 2000a. Dreidimensionale Gebäuderekonstruktion aus digitalen Oberflächenmodellen und Grundrissen. PhD thesis, Universität Stuttgart, Deutsche Geodätische Kommission, DGK Reihe C, Nr. 530.

Brenner, C., 2000b. Towards fully automatic generation of city models. In: IAPRS Vol. 32 Part 3, Amsterdam, pp. 85–92.

de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O., 2000. Computational Geometry: Algorithms and Applications. Springer.

Früh, C. and Zakhor, A., 2003. Constructing 3d city models by merging ground-based and airborne views. IEEE Computer Graphics and Applications, Special Issue Nov/Dec 2003 23(6), pp. 52–61.

Grün, A. and Wang, X., 1998. CC-modeler: A topology generator for 3-D city models. In: D. Fritsch, M. Englich and M. Sester (eds), IAPRS, Vol. 32 Part 4, Stuttgart, pp. 188–196.

Grün, A. and Wang, X., 2001. News from CyberCity-modeler. In: E. Baltsavias, A. Grün, and L. V. Gool (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images (III), Balkema Publishers, pp. 93–101.

Gülch, E. and Müller, H., 2001. New applications of semi-automatic building acquisition. In: E. Baltsavias, A. Grün, and L. V. Gool (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images (III), Balkema Publishers, pp. 103–114.

Gülch, E., Müller, H. and Läbe, T., 1999. Integration of automatic processes into semi-automatic building extraction. In: IAPRS, Vol. 32 Part 3-2W5, München.

Haala, N. and Brenner, C., 1997. Interpretation of urban surface models using 2D building information. In: A. Grün, E. Baltsavias and O. Henricsson (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images (II), Birkhäuser, Basel, pp. 213–222.

Heuel, S., 2002. Statistical Reasoning in Uncertain Projective Geometry for Polyhedral Object Reconstruction. PhD thesis, Rheinische Friedrich-Wilhelms-Universität zu Bonn, Institut für Photogrammetrie (to appear).

OpenSceneGraph, 2004. www.openscenegraph.org/, last accessed 07. April 2004.

Rottensteiner, F. and Briese, C., 2003. Automatic generation of building models from LIDAR data and the integration of aerial images. In: H.-G. Maas, G. Vosselman and A. Streilein (eds), Proc. ISPRS working group III/3 workshop on '3-D reconstruction from airborne laserscanner and InSAR data', Dresden, October, IAPRS Vol. XXXIV, Part 3/W13, pp. 174–180.

Vosselman, G., 1999. Building reconstruction using planar faces in very high density height data. In: in: ISPRS Conference 'Automatic Extraction of GIS Objects from Digital Imagery', Münchem, IAPRS Vol. 32/3-2W5, ISBN 0256- 1840, pp. 87–92.

Vosselman, G. and Dijkman, S., 2001. 3D building model reconstruction from point clouds and ground plans. In: M. A. Hofton (ed.), Proceedings of the ISPRS workshop on Land Surface Mapping and Characterization Using Laser Altimetry, Annapolis, Maryland, IAPRS vol. XXXIV part 3/W4, Commission III., pp. 37–44.

Vosselman, G. and Suveg, I., 2001. Map based building reconstruction from laser data and images. In: E. Baltsavias, A. Grün, and L. V. Gool (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images (III), Balkema Publishers, pp. 231–239.

Weidner, U., 1997. Digital surface models for building extraction. In: A. Grün, E. Baltsavias and O. Henricsson (eds), Automatic Extraction of Man-Made Objects from Aerial and Space Images (II), Birkhäuser, Basel, pp. 193–202.

Wernecke, J., 1994. The Inventor Mentor: Programming Object-Oriented 3-D Graphics with Open Inventor. Addison-Wesley, Reading, Mass.