Efficient and Robust Octree Generation for Implementing Topological Queries for Building Information Models

S. Daum, A. Borrmann
Chair of Computational Modeling and Simulation, Technische Universität München, Germany
daum@bv.tum.de

Abstract. The article presents an efficient and robust algorithm to produce an enhanced octree data structure for topological queries in Building Information Modeling applications. The major benefit of the presented approach is that also poor-quality geometric data can be used as input for the developed spatial query functionality. Additionally, the runtime behavior of the algorithm is optimized in a way that it makes it suitable for rapid queries in a real world scenario. To achieve this, the octree data structure is supplemented with a grid structure which enables fast information propagation from any leaf cell to its neighbors.

1. Introduction

Building Information Modeling is a comprehensive approach for managing the digital information generated and used throughout a building's lifecycle by means of a holistic model. To fulfill this goal, a Building Information Model comprises (1) semantic information of the components and spaces of the building, (2) a detailed description of the geometry of these entities, and (3) the relationships between these entities. Due to the inherent spatial nature of buildings, the spatial relationships play an extraordinary important role.

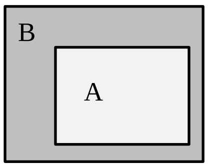
However, the building information models generated by the respective authoring tools in use today include only a small subset of the required spatial relations explicitly. Thus, it becomes desirable to provide means for querying the model for spatial relationships, based directly and solely on the geometric description of the building entities. As a result, the formulation and processing of spatio-semantic queries becomes possible. An example for this kind of query is "Which columns touch ceiling 1?".

The topological operations always correlate two spatial entities. Because of the Boolean return value of an operation, they are also referred to as topological predicates. The operation can be evaluated by the use of the 9-Intersection Model (9-IM) introduced in (Egenhofer, 1991). The 9-IM calculus makes use of the mathematical field of Point Set Topology (Gaal, 1964) where the neighborhood of a point is used to describe topological concepts such as the interior A°, the boundary δA and the exterior A^- of a point set A. The intersection of interior, boundary and exterior of two entities lead to a 3 x 3 matrix whose individual entries are populated with either the empty set symbol or the non-empty set symbol.

$$I = \begin{pmatrix} A^\circ \cap B^\circ & A^\circ \cap \partial B & A^\circ \cap B^- \\ \partial A \cap B^\circ & \partial A \cap \partial B & \partial A \cap B^- \\ A^- \cap B^\circ & A^- \cap \partial B & A^- \cap B^- \end{pmatrix} \tag{1}$$

The resulting matrix can be used to define the topological predicates *disjoint, touch, equals, inside, contains, covers, coveredBy,* and *overlap* (Borrmann & Rank, 2009).

Figure 1 shows the 9-IM matrix for the constellation where object A is *inside* object B and object B *contains* object A, respectively.



$\Rightarrow$

B contains A
A inside B

$$I = \begin{pmatrix} \neg\emptyset & \neg\emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \\ \emptyset & \emptyset & \neg\emptyset \end{pmatrix}$$

Figure 1: Topological Relationship by the 9-Intersection Model

The geometry of the individual components stored in a Building Information Model is usually represented by means of a boundary representation (B-Rep). Since general approaches able to derive topological relationships directly from the boundary representation of the involved entities are not yet known, we are using a space-partitioning representation (octree) as the intermediate geometry format (Borrmann 2007). Consequently, the overall algorithm comprises the followings steps: (1) transform each of the boundary representations of the involved objects into a corresponding octree, (2) create a 9-IM matrix by superimposing the octrees as shown in Figure 2, and (3) determine the applicable topological predicate by comparing the created 9-IM matrix with the predefined ones.

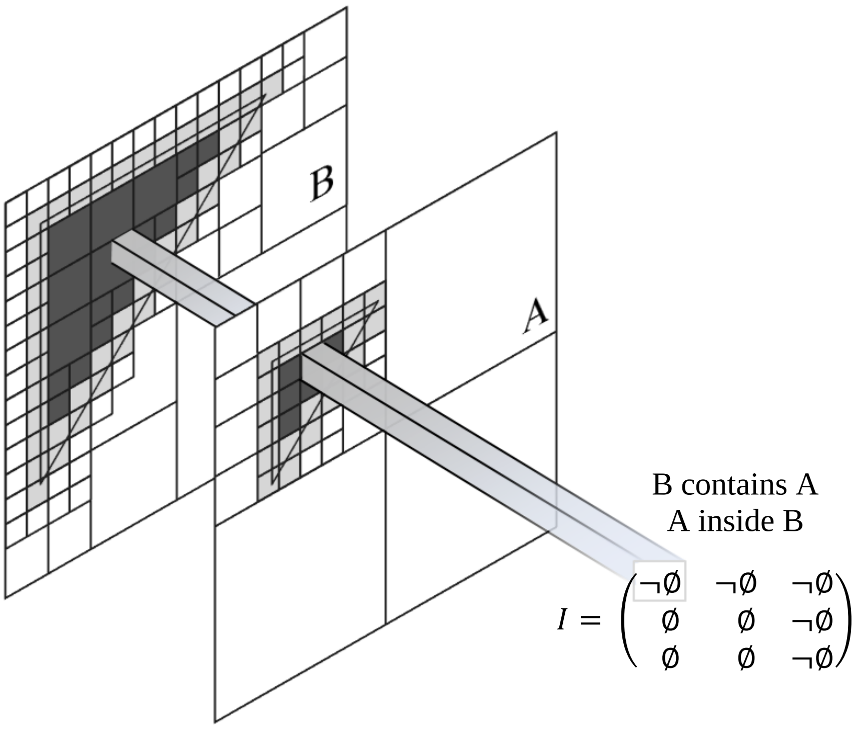


Figure 2: Creating a 9-IM matrix by superimposing two quadtrees

The octree is a hierarchical data structure based on cell decomposition (Samet, 1989). The geometry is decomposed with equally sized cubic cells. To enhance computation performance and to minimize memory allocation needed by the tree, a hierarchical approach is used. Every cell has exactly one parent cell and either 0 or 8 child cells. Cells which are totally inside or outside the represented objects have no child cells. The edge length of a child cell to that of its parent cell is always 1:2. As we decouple a cell with its descendants from the global tree, we *again* obtain an octree. This opens up the possibility to use recursive algorithms.

We use the octree to introduce a layer of geometry abstraction. In addition, the accuracy of the topology predicate can be defined on run time by increasing the refinement level of the tree generation.

2. Related work

The potential benefits of using the functionality of Geographical Information Systems (GIS) for the analysis of dynamical processes in buildings are discussed in (Ozel 2000). The author states that, even if component-oriented CAD systems provide sophisticated functionality for geometric modeling, they normally lack comprehensive spatial analysis capabilities. For this reason, Ozel stores floor plans of buildings in a GIS database in order to use its 2D spatial analysis facilities. Ozel underlines the fact that 3D spatial analysis would be a much more powerful tool for analyzing processes in buildings.

Up to now, spatial database systems that support 3D spatial analysis are only to be found in a research context. The investigations set out in (Gröger et al. 2004), for example, clearly show that the spatial analysis capabilities of the commercial database system Oracle Spatial are limited to 2D space, even though it is possible to store simple 3D geometry.

In the 3D-GIS research community, the main interest lies in the modeling of the ground surface, buildings and infrastructure as well as the subsoil layers. The most important works in this area include (Breunig et al. 1994; Breunig et al. 2001) which report on the development of GeoToolkit, an object-oriented framework for efficiently storing and accessing 3D geographic and geologic data. The main disadvantage of using the framework for analyzing building models is the need to model all spatial entities according to the mathematical concept of simplicial complexes. The obligatory conversion of a boundary representation, as used in CAD tools, to a simplicial complex representation is expensive and, in some special cases, absolutely unfeasible. A more flexible, yet theoretic approach for applying algebraic topology on building models is presented in (Paul and Bradley 2003).

In (Zlatanova et al. 2004; Zlatanova 2006; Coors 2003; Arens et al. 2005) concepts and data structures for storing 3D city models in spatial databases are presented and the suitability of different geometry models for querying topological relationships is discussed. In general, GIS research follows the approach of choosing geometry data structures that implicitly contain topological relationships. Accordingly many of the proposed data structures rely on a simplicial decomposition of the space (Egenhofer et al. 1989; Egenhofer and Herring 1992; Shi et al. 2003). Since building information models are in most case purely geometric representations, we do not assume any pre-defined topological structure in our research.

3. Problem statement

For implementing the spatial operators provided by the query language, algorithms have been developed which are based on an octree representation of the operands (Borrmann 2007).

In the currently available implementation of topological queries, the halfspace-based approach introduced in (Mundani, 2005) is used to generate the octrees corresponding to the geometric objects involved in a topological query (Borrmann & Rank, 2009). For each face of the examined geometric object, two half spaces are created: an interior and an exterior one.

Convex objects can subsequently be represented by the intersection of the half spaces created by their faces. This consideration is used to establish a respective octree generation algorithm.

In reference to the 9-Intersection Model, we define three possible color values for the produced cells in the octree: *White* for a cell which lies in the exterior, *Black* if it is in the interior and *Gray* if the cell is intersected by the boundary of the face. Every gray cell has to be refined until the maximum user defined level is reached. To get to an efficient tree generation, the determinate of a cell's color has to be computed with the less possible effort because of the recursive execution of this function (Figure 3). Mundani (2005) introduced a fast algorithm for the color determination based on the Manhattan Norm. This algorithm is approved to be efficient and robust for convex geometry objects.

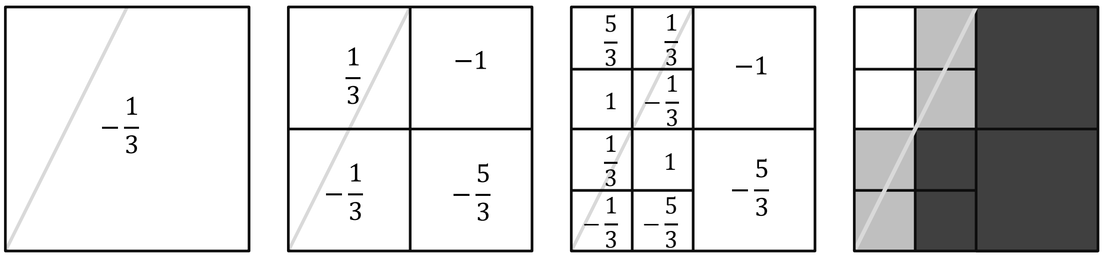


Figure 3: The root cell is intersected by the straight line $a_0 + a_1 \cdot x_1 + a_2 \cdot x_2 = 0$ with $\sum_{i=1}^{3} |a| = 1$. At every refinement a_0 has to be transformed by $\widetilde{a_0} = 2a_0 \pm a_1 \pm a_2$. The signs of the parameters a_1, a_2, and a_3 are defined by the translation direction in space. Illustrated is the refinement for the line equation $-\frac{1}{3} - \frac{2}{3} \cdot x_1 + \frac{1}{3} \cdot x_2 = 0$. Example taken from (Mundani, 2005)

As a concave object is supposed to be transformed into an octree representation using the Mundani approach, it has to be considered as a composition of convex sets of half spaces by combining those using Boolean operations (Figure 4). The required operations are intersection, union and difference.

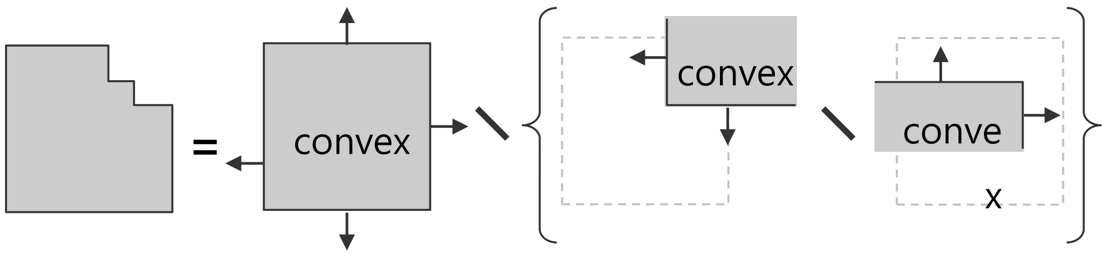


Figure 4: Decomposition of a concave object into its convex parts

The described decomposition into convex parts has to be realized recursively until no new set can be created. The geometry of an object involved in a query is represented as a triangle meshes. Meshes, in generally, consist of vertices which bound edges. These edges again define faces, in our case triangles. The stated components of a mesh have to fulfill certain adjacent criteria in order to represent a valid mesh.

For degenerated meshes, where faces are erroneously not adjacent, the decomposition into convex subparts required for the halfspace-based octree generation is error-prone. Currently available BIM authoring tools often produces such corrupted boundary representations if models are exported (Section 5). To provide spatial query functionality also for such imperfect geometric models, a more robust and efficient implementation of the octree generation has been developed and is presented in the following section.

4. Octree generation

Initially, the fast 3D Triangle-Box Overlap Testing (Akenine-Möller, 2001) is utilized to generate the octree. In the first instance, this tree represents only the boundary of the geometry, but not the interior and the exterior. Consequently, in this state, the tree contains only undetermined and *Gray* colored cells. In the next step, the interior and the exterior is marked by making use of a fill algorithm. To enable filling the octree, the deepest cells of the tree, i.e. the finest octants, are cross-linked such that each octant has direct access to its neighbors. Doing so, the tree is transformed into a combination of a tree/grid data structure. The generated grid is used to efficiently mark the interior cells with a *Black* color attribute. Accordingly, cells located in the exterior are set to *White*. Finally, the tri-colored octree is handed over to the algorithm determining the topological relationship.

4.1 Fast 3D Triangle-Box Overlap Testing

The geometry of the two operands is received as two lists of triangles, which are stored in memory during the execution of the algorithm. By use of this geometry information, the axis aligned bounding boxes of the operands can be calculated. The two bounding boxes are combined to find the appropriated cubic domain which represents both equally sized root octants of the upcoming tree refinement of the examined objects.

Every octant stores a list of integer values. These values represent indices in the triangles list, which are accessible for the octants during execution. The stored indices indicate intersecting triangles for the current octant. If an octant contains intersecting triangles and the maximal tree level is not reached yet, the octant is refined into 8 child octants. The children inherit the indices list from their common parent. The Triangle-Box Overlap test is started again for each newly created octant. If a considered triangle does not intersect with the octant, the corresponding index is deleted from the octant's integer list. This approach ensures that only potential intersecting triangles are tested against the current octant. Furthermore, the memory expensive triangle data only needs to be retained once. An octant is not furthermore refined if its indices list is empty or the maximal refinement level is reached.

The implementation of the Triangle-Box Overlap test follows the Hyperplane Separation Theorem based algorithm presented in (Akenine-Möller, 2001). According to this theorem, two convex geometry objects are disjoint, if

- there is a separating axis parallel to the normal of a face of either of the objects

or

- there is a separating axis along an axis obtained by the cross product of an edge of the first object and an edge of the second one.

To deduce if a triangle intersects a box, 13 axes have to be tested against. In the presented case, the box represents an octant and is therefore a cube, which leads to an even more efficient implementation.

As the routine only involves efficient subparts like computing the cross product of vectors and interval overlay checking, it is faster than face based tests, e.g. intersecting the triangle with all faces of the box.

4.2 Cross-linking of leaf cells in the octree

At this point of the algorithm, the tree comprises *Gray* marked cells, indicating intersections by the boundary of the geometry. The remaining cells are still undetermined in their color. These cells are either located in the interior or the exterior of the bounding geometry of the operand. Thus, the two uncolored cell groups are separated by the *Gray* cells between them. This makes the use of a flooding algorithm which sets the color attribute of the cells applicable if it uses the *Gray* cells as boundary. To efficiently mark the leaf cells according to their location, each cell has to detect its neighbors. Crouse (2003) presents an algorithm for finding the neighbors of a cell by use of a bit code for the local locations of cells in their common parent (Figure 5).

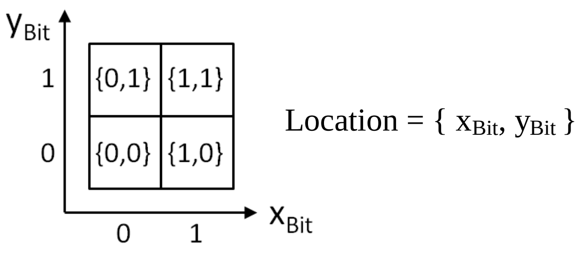


Figure 5: Bit code of the location of cells in their parent cell in 2D

If the desired neighbor of a leaf cell is located within the same parent cell, its identification is trivial. Otherwise, to find neighboring cells in different parents a search is performed. To this end, a *Path* is defined as a sequence of locations $\{\{x_{Bit}, y_{Bit}\}_{Level}\}$. If the tree is traversed from a leaf cell in order to get outside lying neighbors, the *Path* data structure is repeatedly filled with the current location bits until the bit of the examined direction changes. This is depictures in the top of figure 6. Note that in the first *Path* structure the x_{Bit} in the last position has switched from 1 to 0. Accordingly, a neighbor in the investigated direction lies in the currently reached level. To get possibly existing adjacent child cells, the *Path* data is modified. All bits in the respective direction are inverted and the ordering of locations is reversed. By this altered structure, we travel down the tree from the determined cell which contains the examined cell and the neighboring cell in the desired direction. The traversal is done as deep as possible.

Thus, the direct neighbors of the initially examined cell are reached, even if they are located in another parent cell (Figure 6).

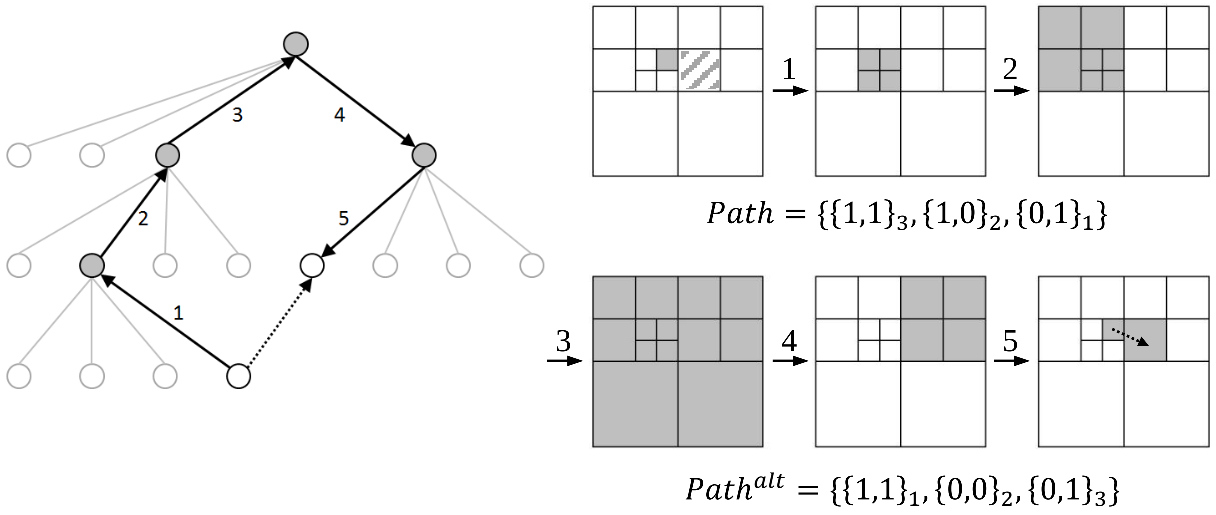

1
2
3
4
5
1
2
3
4
5
$Path = \{\{1,1\}_3, \{1,0\}_2, \{0,1\}_1\}$
$Path^{alt} = \{\{1,1\}_1, \{0,0\}_2, \{0,1\}_3\}$

Figure 6: Finding adjacent cells by tree traversals routed via bit encoding for cell locations

In (Crouse, 2003), the octree is smoothed such that a maximum difference of one level for neighboring cells is achieved. Since this prevents the full advantages of the hierarchical tree structure concerning memory allocation, we disregard this smoothing step. As a consequence, a cell can have an infinite count of neighbors in theory. In practice, however, the count of neighboring cell is still manageable. Therefore, adjacent cells can be stored in a list structure in each cell. The knowledge of neighboring cells is ideal for fast information transfer between adjacent cells. The flooding algorithm described in the following section makes use of this optimization.

4.3 Flooding the leaf cells of the octree

The presented flooding algorithm targets the classification of interior and exterior cells. In the first step, an arbitrary cell which is not marked *Gray* is picked, and an inside/outside test is performed for it. This is repeated until at least one interior and one exterior cell is found.

In building information models in use today there are only very rare occurrences of geometric objects with caves. Therefore we regard only caveless geometry here. This allows the application of robust inside/outside tests, using intersection rays, for example, and eases the application of the flooding algorithm. Please note that spaces, which are important operands for topological queries, are regarded as distinct volumetric objects, for which in most cases, the assumption of cavelessness holds.

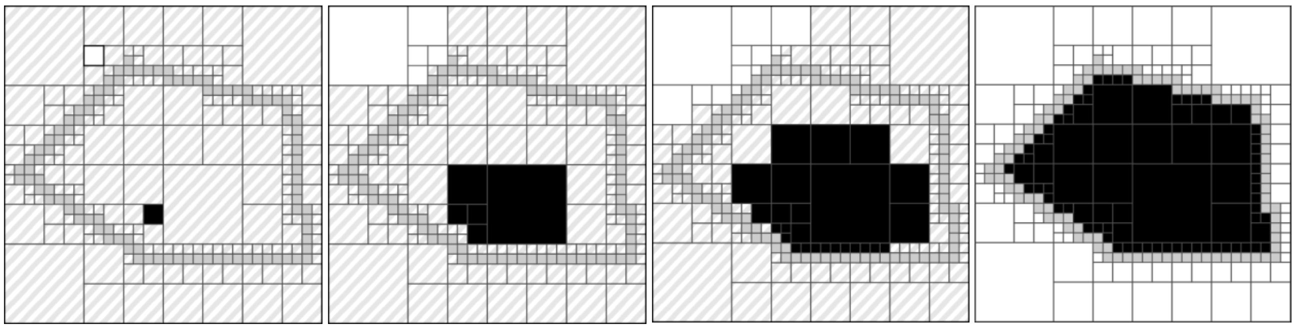


Figure 7: Information transfer by the flooding algorithm in 2D

As soon as a first interior cell is found, its adjacent neighbors in the six axis directions are marked as *Black* except for those which have previously been set to *Gray*. This procedure is recursively called for the found colorless neighbors, which establishes a flooding of the *Black* color to all interior cells. Likewise, the exterior cells are marked as *White* (Figure 7).

5. Implementation of topological operators

The algorithm determining the topological relationship between two operands uses the generated three-colored octrees as input. On each recursion level, pairs of octants are created with one octant originating from object A and one octant from object B, both representing the same sector of the 3D space. Each octant pair provides a color combination to which specific rules can be applied. These rules may lead to filling a 9-IM working matrix that is maintained by the algorithm to keep track of the knowledge gained about the topological constellation. There are 12 positive and 9 negative rules altogether. A positive rule (Figure 8) can be applied when a certain color combination occurs, and a negative rule (Figure 9) if certain color combinations do not occur over an entire level. Positive rules lead to empty set entries in the matrix, negative rules to non-empty set entries.

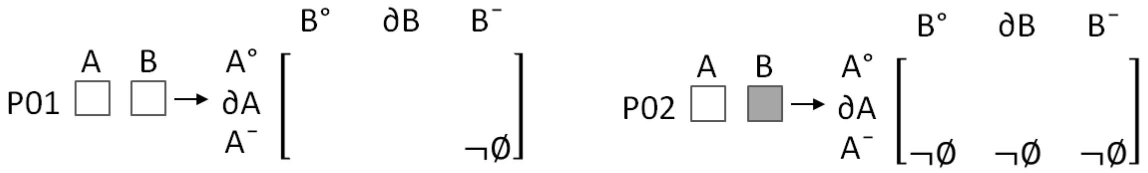


Figure 8: Examples for Positive Rules.

The rules are derived from the semantics of the colors. If a *White* octant of the first operand occurs at the same place as a *Black* octant of the second operand, it follows that the intersection between the exterior and the interior of the operands is non-empty. The 9-IM working matrix is successively filled by applying these rules to all octant pairs. When processing the two operators against each other to obtain the accurate topological attribute, the working matrix is compared with all predicate matrices of the formal definitions: *touch, equals, inside, contains, covers, coveredBy,* and *overlap.* If the working matrix complies fully with one of them, the recursion is aborted and the algorithm returns the respective predicate. If there is any contradiction between the filled matrix and the matrix of a predicate, the respective predicate is precluded. If no unequivocal decision is possible for any of the predicates, a further refinement is necessary, i.e. octant pairs of the next level are created.

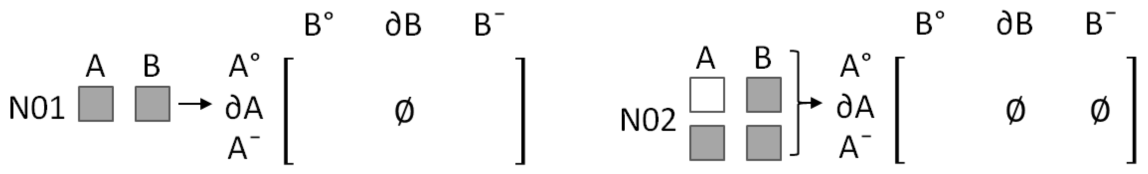


Figure 9: Examples for Negative Rules

In the case of the predicate operators the 9-IM working matrix is checked against the corresponding predicate matrix only. If there is a contradiction the algorithm returns false, if it completely complies, it returns true. If, after execution of all applicable rules, the current occupancy of the working matrix does not allow for validation or disproval of the/any predicate and the maximum refinement level is not reached, child pairs are created and the algorithm calls itself recursively. If the algorithm reaches the maximum refinement level and, the result is not determinable, a so-called predicate hierarchy is applied, which again ensures that the most probable situation is detected.

6. Valid tree generation despite complex geometry

The previously used half space oriented method (Mundani, 2005) can exclusively operate on convex geometry. If concave geometry is involved in a query, it has to be decomposed to its convex subparts. The subparts represent the same geometry if they are combined in certain order by use of Boolean operations on half spaces. Firstly, this entails that the tree generation is executed repeatedly and the individual results have to be combined as well, which leads to a multiplication of the runtime. Secondly, with increasing complexity of the geometry, the count of involved half spaces representing the geometry increases accordingly. If one single half space is set up erroneous, the entire octree is corrupted because the half spaces interact globally with each other. On the other hand, the presented algorithm solely creates the octree's subparts by local data. So, the result does not have dependencies from distantly located geometry parts. Furthermore, this approach reacts, to a certain extent, uncritically to imperfect input geometry, which is often created by available BIM software.

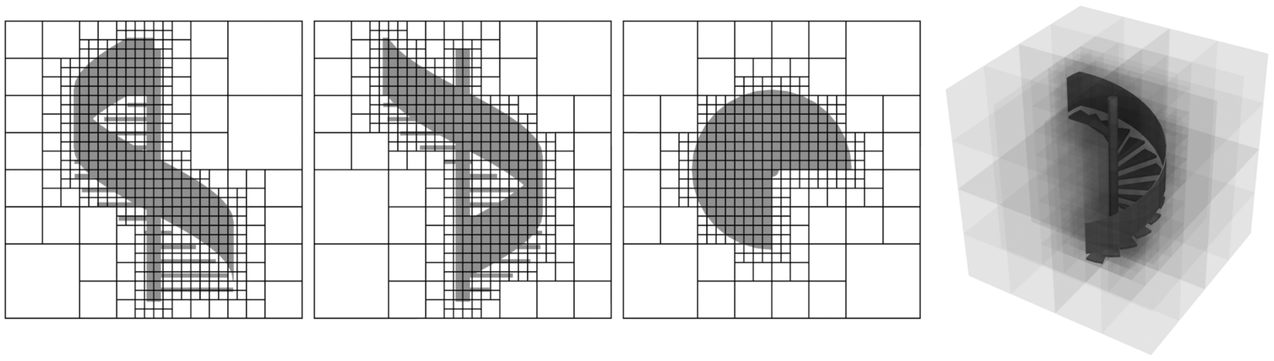


Figure 10: Complex B-Rep geometry (e.g. a circular staircase) transferred to the octree data structure by use of the presented algorithm (f.l.t.r: Front-, Right-, Top-, 3D-View)

7. Conclusion

The paper presents an efficient and robust method to produce octree structures from B-Rep geometry in BIM. The octree generation method has been successfully integrated in a spatial query language (Borrmann, 2007). The algorithm determining the topological relationship uses the octree representations of the operands as input. Therefore, the accuracy of the trees is essential to deduce valid results for the topological queries. The necessary accuracy is delivered by the described octree generation method and verified with real world data used in the construction industry. Furthermore, the user of the spatial query language experiences shorter execution times by use of the described method. This is an advantage over the half space based approach (Mundani, 2005), used in the previous implementation. The acceleration is achieved by the faster octree generation and the enhanced data structures that make a rapid information transfer between adjacent octants possible. Additionally, the stated method is more robust, as it does not contain a decomposition of concave geometry in its convex subpart as the Mundani algorithm. The robustness is also achieved with degenerated input geometry, which is often produced and used by BIM authoring tools.

References

Akenine-Möller, T., (2002). *Fast 3D triangle-box overlap testing*. J. Graph. Tools 6(1), pp. 29-33.

Arens, C., Stoter, J., van Oosterom, P., (2005). *Modelling 3D spatial objects in a geo-DBMS using a 3D primitive.* Computers & Geosciences, 31(2), pp. 165–177.

Borrmann, A., (2007). *Computerunterstützung verteilt-kooperativer Bauplanung durch Integration interaktiver Simulationen und räumlicher Datenbanken.* Dissertation, Technische Universität München.

Borrmann, A., Hyvärinen, J. & Rank, E., (2009). *Spatial constraints in collaborative design processes.*, In: Proc. of the Int. Conf. on Intelligent Computing in Engineering (ICE'09). Berlin, Germany.

Borrmann, A., Rank, E. (2009). *Topological analysis of 3D building models using a spatial query language*, Advanced Engineering Informatics 23(4). pp. 370-385.

Borrmann, A., Rank, E. (2009). *Query Support for BIMs Using Semantic and Spatial Conditions*. In: Underwood. J. and Isikdag. U. (Eds): Handbook of Research on Building Information Modeling and Construction Informatics: Concepts and Technologies, IGI Global.

Breunig, M., Bode, T., Cremers, A., (1994). *Implementation of elementary geometric database operations for a 3D-GIS*. In: Proc. of the 6th Int. Symp. on Spatial Data Handling.

Breunig, M., Cremers, A., Müller, W., Siebeck, J., (2001). *New methods for topological clustering and spatial access in object-oriented 3D databases*. In: Proc. of the 9th ACM Int. Symp. on Advances in Geographic Information Systems.

Coors, V., (2003). *3D-GIS in networking environments.* Computers, Environment and Urban Systems, 27(4), pp. 345–357.

Crouse, B., (2003). *Lattice-Boltzmann Strömungssimulationen auf Baumdatenstrukturen.* Dissertation, Technische Universität München.

Egenhofer, M., Herring, J., (1989). *A mathematical framework for the definition of topological relationships.* In: Proc. of the 4th Int. Symp. on Spatial Data Handling.

Egenhofer, M. (1991)., *Reasoning about Binary Topological Relations.* In: Proc. of the 2nd Symp. on Advances in Spatial Databases (SSD'91).

Gaal, S. (1964). *Point Set Topology*. Academic Press.

Gröger, G., Reuter, M., Plümer, L., (2004). *Representation of a 3-D city model in spatial object-relational databases*. In Proc. of the 20th ISPRS congress.

Mundani, R.-P., (2005). *Hierarchische Geometriemodelle zur Einbettung verteilter Simulationsaufgaben.* Dissertation, Universität Stuttgart.

Ozel, F., (2000). *Spatial databases and the analysis of dynamic processes in buildings*. In: Proc. of the 5th Conf. on Computer Aided Architectural Design Research in Asia.

Paul, N., Bradley, P. E., (2003). *Topological houses*. In: Proc. of the 16th Int. Conf. of Computer Science and Mathematics in Architecture and Civil Engineering.

Shi, W., Yang, B., Li, Q., (2003). *An object-oriented data model for complex objects in three-dimensional geographical information systems.* Int. J. of Geographical Information Science, 17(5), 411–430.

Samet, H., (1989). *Applications of Spatial Data Structures: Computer Graphics, Image Processing and GIS.*:Addison-Wesley.

Zlatanova, S. (2006). *3D geometries in spatial DBMS*. In: Proc. of the Int. Workshop on 3D Geoinformation 2006.

Zlatanova, S., Rahman, A., Shi, W. (2004). *Topological models and frameworks for 3D spatial objects*. Journal of Computers & Geosciences, 30(4), 419–428.