Automated generation of building fingerprints using a spatio-semantic query language for building information models

S. Daum & A. Borrmann,
Chair of Computational Modeling and Simulation

C. Langenhan & F. Petzold
Chair for Architecture Informatics

Leonhard Obermeyer Center, Technische Universität München

ABSTRACT: In early stages of building design, architects frequently make use of referential solutions as a source of inspiration. Today, these reference solutions are looked up manually in a time-consuming and laborious process. To speed up the process, means for automating this procedure are required. The approach discussed here is based on the assumption that the reference solutions (i.e. building designs) are stored in a repository as building information models (BIM) using the open data model Industry Foundation Classes (IFC). In order to find suitable reference solution for a given problem, a measure of similarity has to be defined. To this end, we introduce the notion of the building fingerprint as a way to capture the main characteristics of a building design. A major component of the fingerprint is the representation of the adjacency relationships between the spaces, which can be expressed by an adjacency graph. Another component is the accessibility relationships, which again can be expressed by a corresponding graph. To use this graphs as a basis for the lookup mechanisms, they have to be generated in an automated way for the large set of IFC models stored in the reference solution repository. In this paper, we present an approach based on the BIM query language QL4BIM which provides high-level topological operators to efficiently retrieve accessibility and adjacency relationships among spaces. We discuss how QL4BIM is applied to generate the graph components of the building fingerprints.

1 INTRODUCTION

The complexity of the requirements on buildings is continuously increasing and thus often confronting designers with interdisciplinary problems, reaching far beyond the traditional challenges and methods of architecture and engineering. The iterative nature of the design process results in a continuous exchange between creative, analytical and evaluation stages to select the most promising design variations. Using already built or designed buildings as a reference helps to evaluate the results of a design process and supply guidance through design problems.

Most of the computational search methods available today rely on textual rather than graphical approaches to represent information. For spatial configurations such as floor plans, however, textual descriptions tend to be insufficiently precise. To overcome these shortcomings, Langenhan et al. (2013) introduced a novel approach which allows an automated lookup of reference solutions from a repository using graphical search keys. The repository is made up of a large set of building information models (BIM) described by the neutral data model Industry Foundation Classes (IFC, ISO 16739:2013).

For specifying the search key, the notion of the Building Fingerprint was introduced which captures the main characteristics of a building's design. It forms the basis of assessing the similarity of an individual reference solution with the specified problem. Accordingly, the Building Fingerprint is applied as index of the building model repository.

A major component of the fingerprint is the representation of the adjacency relationships between the spaces, which can be expressed by an adjacency graph. Another important component is the accessibility relationships among the spaces, which also can be expressed by a corresponding graph. To use this graphs as a basis for the lookup mechanisms (the index), they have to be generated in an automated way for the large set of IFC models stored in the reference solution repository.

In this paper, we present an approach based on the BIM query language QL4BIM which provides high-level topological operators to efficiently retrieve accessibility and adjacency relationships among spaces (Daum and Borrmann, 2013a). We discuss how QL4BIM is applied to generate both the accessibility graph and the adjacency graph required to generate expressive building fingerprints. The approach combines semantics- and geometry-based analysis and filtering capabilities provided by QL4BIM.

2 PROBLEM DESCRIPTION

The objective of the presented research is to provide the building designer with relevant information that can serve as reference knowledge for the current planning task and as inspiration for the building design, as well as to provide a structured and targeted means of organizing existing information about buildings. For this it is necessary to define requirements with regard to the data structure employed and corresponding methods for retrieving and accessing this data. In this paper, we focus on topological and functional information and consider the individual storeys separately.

As accessibility and adjacency graphs are not explicitly represented in an IFC model, they have to be derived from the available entities. In principle, this can be achieved by navigating the semantic part of the data model and gathering the required relationship information. However, due to incomplete or erroneous implementation of the IFC, which is found in many export modules of BIM authoring applications, the required relationship information is frequently not available in the exported models. In this case, advanced geometric processing is necessary to obtain the desired graphs from pure geometric information.

This can be achieved by means of QL4BIM. In contrast to other BIM query languages, QL4BIM allows to analyze a BIM directly on the basis of pure geometric information. It is capable to derive topological relationships between building elements. For the task of Building Fingerprint extraction from IFC model, these topological operators have been extended and supporting functions have been included.

2.1 *BUILDING FINGERPRINTS*

Current electronic search methods use textual information rather than graphical information. The configuration of space and the relations between them are hard to represent using keywords, resulting in imprecise descriptions of architecture.

Inspired by the principle that a person can be identified through his/her fingerprints, so-called Building Fingerprints (Langenhan et al., 2013) of a building are created that combine metadata concerning the different parameters of a building.

This includes spatial properties, such as how rooms are connected. These topological relationships can be represented as graphs and compared using computational methods. The process of querying a larger floor plan repository can be interpreted as formulating a query as a sub-graph and search for this structure in the repository (Langenhan et al. 2013).

Adapting the way designers work in the early design stages, we use a sketch-based query specification with which it is possible to submit imprecise or fuzzy queries that may be vaguely formulated or that represent only part of the problem. In the proposed approach, the user sketches spaces as boxes and uses straight lines to define relationships between the spaces. A single line indicates an adjacency relationship between spaces, whereas a double line specifies an accessibility relationship. Figure 2 gives an example of a sketched query with a bath, kitchen and living room. The sketch is transformed into a query graph which is subsequently used to query the graph repository. The graph repository is built up by analyzing the BIM repository (Langenhan et al. 2013).

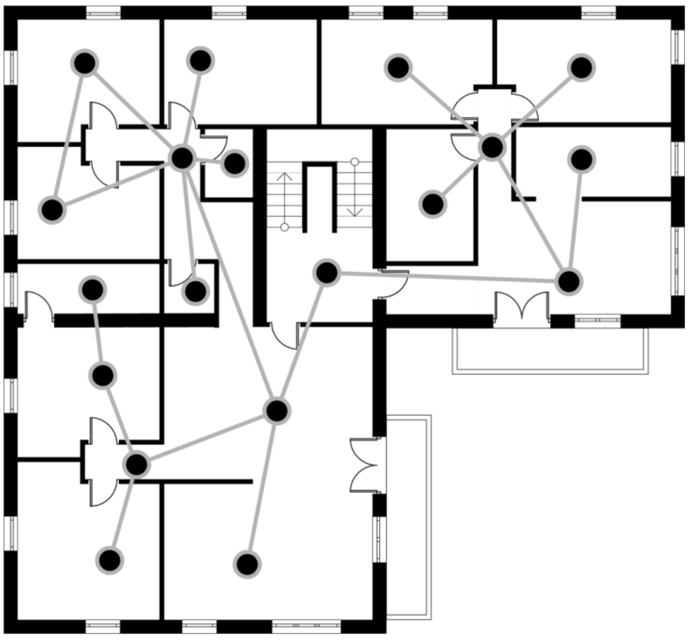


Figure 1: Floor plan of residential building overlaid by adjacent graph visualization.

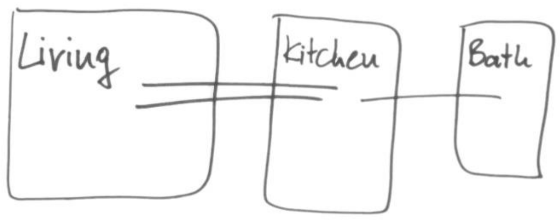


Figure 2: Sketch-based query to derive a query graph for the subgraph matching.

2.2 *IFC-based adjacency/accessibility modeling*

Building Information Models are comprehensive digital representations of buildings. The data model Industry Foundation Classes (IFC) is a neutral data format which provides means for representing and exchanging these models. The IFC data model incorporates a wide set of classes to describe a building including its components, the enclosed spaces and their mutual relationships (Weise et al., 2009)

To represent relationships between entities, relationship classes are used. As these links are represented by dedicated object instances rather than simple references, they are called objectified relationships. In addition to the semantic information, an IFC model includes the geometry of building elements and space objects.

In principle, the IFC model is able to represent information regarding the adjacency and accessibility of rooms. However, a rather complex data structure is employed. To extract a global adjacency / accessibility graph, fine-granular semantic information has to be gathered and combined as discussed in the following sections.

2.2.1 *Adjacency*

Figure 3 shows a section of the semantic part of the IFC model which is used to express adjacency relations. This includes the classes *IfcSpace*, *IfcRelSpaceBoundary*, *IfcConnectionGeometry* and *IfcWall / IfcWallStandardCase*. The *IfcSpace* class represents the inner volume of a room. The class *IfcRelSpaceBoundary* links a room to one of its bounding building elements, e.g. a wall. The *IfcConnectionGeometry* specifies the area of contact between the space and the building element precisely.

If the semantic and geometrical modelling is of high quality, the adjacency of two rooms can be determined by finding two *IfcSpaces* which refer to the same *IfcWallStandardCase* (Figure 3). However, to avoid erroneous detection of adjacent rooms, walls have to be separated at connection points.

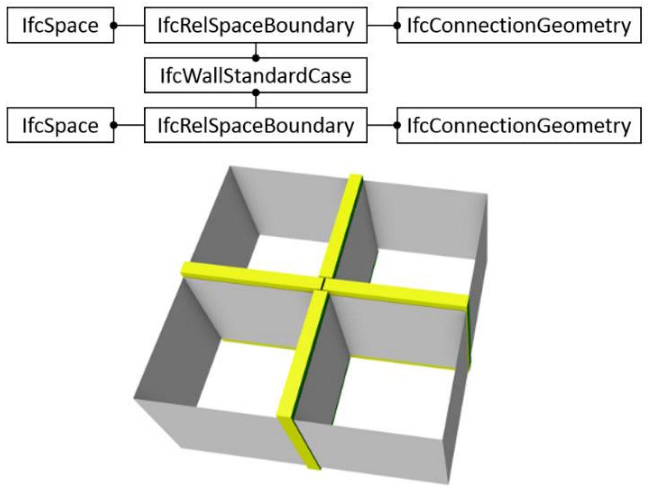


Figure 3: Correct semantic and geometrical modelling of rooms and their relationships in the IFC data model

2.2.2 *Accessibility*

Rooms are mutually accessible if they share a wall with a door. The correct constellation of IFC entities is depicted in Figure 4.

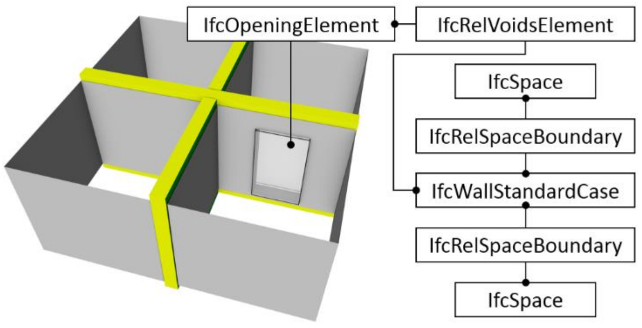


Figure 4: Retrieving accessibility information from the IFC model by openings between rooms.

2.2.3 *Issues related to pure semantic analysis*

As described in Langenhan et al. (2013), it is possible to extract adjacency/accessibility graphs from IFC models using exclusively semantic information. However, the described method is heavily dependent on a correct semantic description of the model.

This may cause severe problems as the part of an IFC model which characterizes adjacency / accessibility relationships is frequently defective or missing completely. Reasons for these quality issues are modelling errors and/or imperfect export mechanism of the BIM tool employed. To overcome these issues, Section 4 presents alternative approaches to extract Building Fingerprints which rely on a significantly reduced set of semantic informations.

3 QL4BIM – A BIM QUERY LANGUAGE

3.1 *Analysis and filtering of BIMs using QL4BIM*

QL4BIM is a comprehensive query language for IFC-based building information models which allows not only the navigation of complex object networks and the formulation of expressive query statements, but in particular provides spatial operators which allow a high-level analysis of qualitative spatial relationships of the physical entities and spaces comprised by the BIM (Daum and Borrmann, 2013a).

To analyze a model, QL4BIM operates on user-defined collections of IFC entities. E.g. a collection can be established containing all *IfcProduct* entities. Subsequently, the collection can be filtered by applying a specified predicate as filter conditions.

For example, a subset of the *IfcProducts* collection can be created which contains only *IfcWall* objects (a sub-class of *IfcProduct*) which have a height greater than 2.5 meters.

3.2 *Qualitative topological operators of QL4BIM*

QL4BIM provides eight topological predicates: *Disjoint*, *Equal*, *Touching*, *Containing*, *Inside-of*, *Covering*, *Covered-by*, and *Overlapping* (Figure 5).

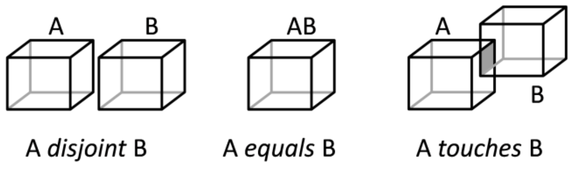


Figure 5: Three of the eight topological predicates of QL4BIM.

These predicates correlate two spatial entities and can be described by the 9-Intersection Model (9-IM) introduced in Egenhofer (1991). The algorithms implementing the topological operators of QL4BIM operate directly on the BRep geometry of components and spatial containers provided by the IFC model. They are designed to be applied on large datasets and achieve very high performance (Daum and Borrmann, 2013c).

However, the computational extraction of Building Fingerprints on a pure qualitative topological examination turned out to be error-prone because of false positive classifications of adjacency / accessibility relationships. Therefore, we extended the topological operators to return values about the degree of interaction between the operands.

3.3 *Enhanced quantitative topological operators bases on octree structures*

In order to realize a robust extraction of Building Fingerprints from BIMs, the topological operators use an octree-based implementation method. The octree-based method enables that information concerning the degree of interaction between two spatial entities can be determined. For example, the volume of an intersection or the area of a touch constellation can be exploited. This quantitative analysis of topological relationships allows us to distinguish constellations in which small parts of building elements interact geometrically from those with large areas of interaction.

For evaluating topological relationships, the octree approach is more expensive than the direct use of the BRep data structure of entities (Daum & Borrmann 2014). However, the correctness of the extracted fingerprint is crucial for the downstream process of matching the sketched user input to a solution candidate. Additionally, the extraction of adjacency/accessibility graphs is achievable with a coarse discretization of geometry, which drastically improves the runtime performance of the octree approach.

3.3.1 *The Octree data structure*

The octree is a hierarchical data structure based on cell decomposition (Samet, 1989). The geometry is decomposed by cubes, named octants in this scope. The differently sized octants are arranged in a hierarchical manner which forms a tree structure, shown in 2D in
Figure 6.

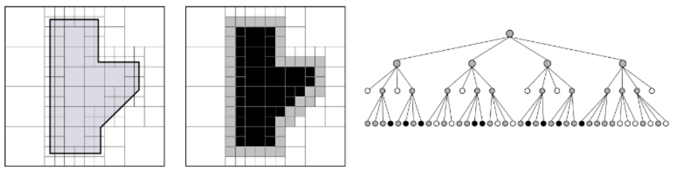


Figure 6: A non-colored quadtree and a three-colored quadtree with its graph structure.

To transfer the BRep geometry of building elements and spatial containers to octrees, the hyperplane-based box/triangle test (Akenine-Möller, 2001) is utilized. After the recursive refinement based on the computed box/triangle intersections, the tree represents the boundary of the geometry (*Gray* cells), whereas the interior and the exterior are not classified. For classifying the interior and the exterior cells we apply a flooding algorithm which starts at an arbitrary interior / exterior cell and propagates trough the unclassified areas.

To enable efficient flooding of the octree, we enhanced the octree data structure: The leaves of the tree 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*. The flooding of the tree to generate the three colored data structure is shown by its 2D equivalent in Figure 7.

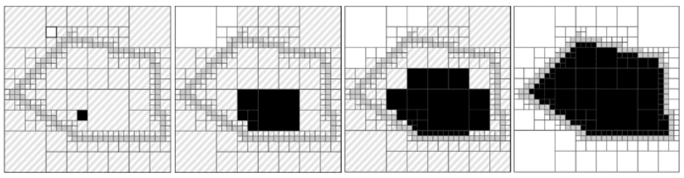


Figure 7: Efficient information propagation between neighboring octants by the developed tree/grid data structure in 2D.

3.3.2 *The quantitative topological operators*

The extraction of Building Fingerprints from BIMs requires additional information of the interaction degree between building elements and spaces. To allow direct access to this information, the range of topological functions of QL4BIM was extended: The separated groups of octants representing the exterior, the boundary and the interior of an object can be tested for intersection against a second group. For example, the number of octants between the boundaries of two *IfcSpaces* can be determined.

We denote objects which separate the actual analyzed objects as secondary objects. For example, a wall which separates two spaces is a secondary object, whereas the spaces themselves are primary objects. In our approach to adjacency detection, we neglect the secondary objects and test the primary objects on a touch constellation.

To realize this, each octant in the groups under test is scaled in place to extend the outer hull defined by all octants. Figure 8 shows this approach for two octants contained in two separated octant groups.

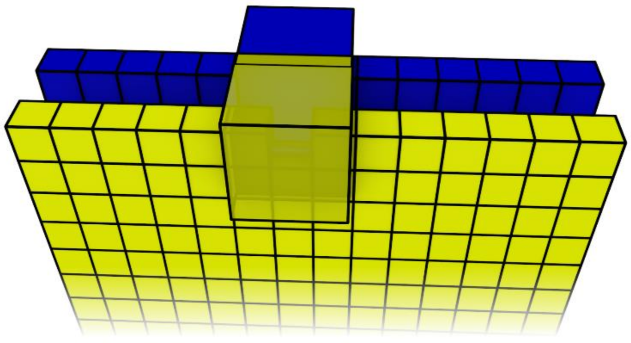


Figure 8: In place scaling of octants to overcome gaps

Then the number of intersecting octant pairs is computed. This allows to distinguish cases where a low number of octants overlaps (a corner configuration) from cases where a high number overlaps (adjacency along a wall). A threshold can be specified to define when two spaces are classified as adjacent. A true adjacency relationship and a corner configuration between two spaces is illustrated in Figure 9.

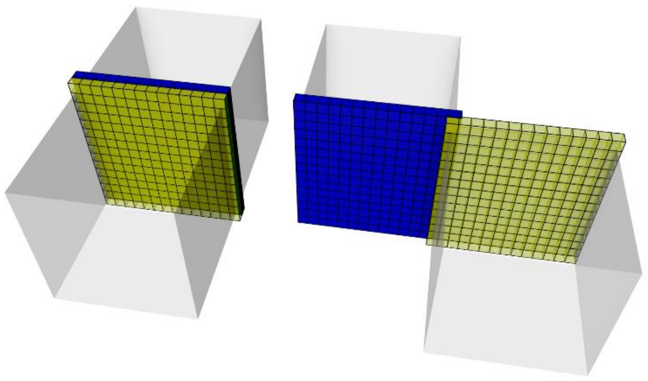
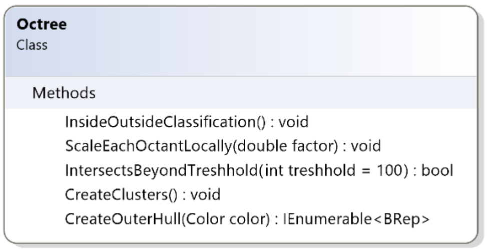


Figure 9: Voxelization of the connection geometry between two spaces. A true adjacency relationship (left) and a corner configuration (right) can be distinguish by counting intersecting octants.

3.3.3 *Additional functionality and its deployment*

The enhanced quantitative topological operators and additional functionality is made available through methods of the octree data structure. Figure 10 depictures these methods.


Octree
Class

Methods
InsideOutsideClassification() : void
ScaleEachOctantLocally(double factor) : void
IntersectsBeyondTreshhold(int treshhold = 100) : bool
CreateClusters() : void
CreateOuterHull(Color color) : IEnumerable<BRep>

Figure 10: Enhanced quantitative topological processing and supporting function provided by the octree class.

The method *InsideOutsideClassification* uses the flooding approach described in Section 3.3.1 to classify each of the octants as being located either inside, on-boundary or outside. *ScaleEachOctantLocally* increase the size of each octant in the tree to close gaps produced by secondary building elements. This was already visualized in Figure 8.

The Boolean method *IntersectsBeyondTreshold* returns *true* if the number of intersecting octants of two specified groups (*White*, *Gray*, or *Black*) is greater than the specified threshold. *CreateClusters* searches for octant clusters with the same color. A cluster is a connected group of octants of the same color. The implementation of this method utilize the tree's added grid functionality as efficient method to traverse to neighboring octants. The grid functionality also supports identifying faces between different colored clusters. This can be used to create hulls for clusters of a specific color. This functionality is available through the *CreateOuterHull* method which returns a 3D BRep.

3.4 *Live LINQ*

The QL4BIM system is based on the object-oriented declarative query language LINQ (Meijer et al., 2006) for evaluating query expressions on collection level. In LINQ queries, the attributes and methods of involved objects can be used. The developed additional functionality discussed in Section 3.3.3 is therefore provided as methods of the class *Octree*.

In the standard LINQ implementation queries have to be established at compile time. To provide a more flexible system, the Live LINQ system is introduced in which filter statements can be stated during runtime. The enhancements applied to evaluate LINQ to Live LINQ are is described in (Daum and Borrmann, 2013b).

An example of a query is shown in Figure 11, where all *IfcProducts* are selected which overlap wall1 and which are located above slab1.

```
IfcProducts.Where(p => { var wall1 = IfcProducts [25];
                          var slab1 = IfcProducts [12];
                          return p.Overlaps(wall1) && p.Above(slab1); })
```

Figure 11: Live LINQ QL4BIM query

4 QL4BIM QUERIES FOR THE EXTRACTION OF BUILDING FINGERPRINTS

In the following section, three QL4BIM methods will be presented which allow to extract adjacency graphs from an IFC model. Each of the approaches assumes less semantic information being provided by the IFC model than its predecessor. Finally, a purely geometry based method to determine the accessibility relations between rooms is explained.

4.1 *Semantic-based adjacency extraction from high quality IFC models*

The first extraction method assumes that all required semantic information is present in the model. This involves the classes *IfcSpace*, *IfcRelSpaceBoundary*, *IfcConnectionGeometry* and *IfcWallStandardCase*. Two rooms can be identified as adjacent if two *IfcSpace* entities refer to the same *IfcWallStandardCase* using the *IfcRelSpaceBoundary* relationship

Figure 12 shows the QL4BIM query to extract the adjacency relationships by use of the described semantic model. The query makes use of a collection filled by all *IfcProduct* entities of the current model.

```
var decollators = IfcProducts.Where(p => p.ProvidesBoundaries.Count == 2);
return decollators.Select(p => {
                    var space1 = p.ProvidesBoundaries[0].RelatingSpace;
                    var space2 = p.ProvidesBoundaries[1].RelatingSpace;
                    return new Pair(space1, space2);});
```

Figure 12: Adjacency extraction based purely on the semantic model (QL4BIM query)

The *ProvidesBoundaries* property yields a list of all *IfcRelSpaceBoundary* objects from which the current *IfcProduct* is referenced. If this list comprises two items, the current product is added to the temporal variable *decollators*. In the next step, pairs of *IfcSpaces* are created by following the two *IfcRelSpaceBoundary* entities referenced by one decollator. If the IFC instance is correctly modeled, the returned enumeration includes pairs of adjacent rooms.

4.2 *Semantic-based adjacency extraction from defective IFC models*

Due to variations in BIM creation (modeling), deviant space-wall constellations may appear. For example, a wall can separate more than two pairs of rooms (Figure 13). In this case, the method described in Section 4.1 does not work properly, as pairs of *IfcSpaces* without actual adjacency configuration might be created. To overcome this issue, the two *IfcConnectionGeometry* instances are used to verify the possible adjacency. This geometry indicates the exact position of the touching area between an *IfcSpace* and the building element. In the first step, both geometry objects are transferred to three-colored octrees.

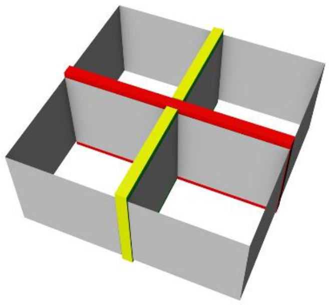


Figure 13: Imprecise IFC modelling concerning room definition caused by a continued wall.

In the following, only the *Gray* octants of both octrees are considered as they represent the discretized volumes where the connection geometry is located. As this surface-based geometry has been voxelized, it now has a voluminous extension. To verify the adjacency between the two spaces under test, the intersection between the two gray octant groups is investigated. As described in Section 3.3.3, we close the gaps between the connection geometry on both sides of the wall by a local scaling of the octants. The number of intersecting octants is used to assess the geometrical constellation of the two connection geometry items, preventing false positive adjacency classifications. The approach is depicted in Figure 14.

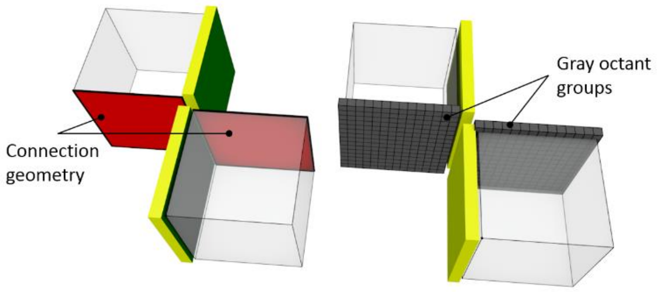


Figure 14: Disprove an adjacency constellation between two rooms by examine the IfcConnectionGeometry

4.3 *Adjacency extraction by IfcSpace geometry*

The third extraction method can be employed if the IFC model contains *IfcSpaces* with linked 3D geometry representing room shapes, but *IfcSpaceBoundary* and *IfcConnectionGeometry* objects are not present. In this case, the *IfcSpace* geometry is directly used to determine the adjacency of rooms. The resulting QL4BIM query is depicted in Figure 15.

```
1 var octrees = IfcSpaces.Select(s => s.CreateOctreeFromBRep);
2 foreach (var octree in octrees) {
3           octree.InsideOutsideClassification());
4           octree.ScaleEachOctantLocally(1.8));}
5 foreach (var octree1 in octrees)
6           foreach (var octree2 in octrees)
7                     if(octree1.IntersectsBeyondTreshold(octree2, Color.Gray))
8                               listOfPairs.Add(new Pair(octree1.Id, octree2.Id);
9 return listOfPairs;
```

Figure 15: IfcSpace-based adjacency extraction

The query comprises the following steps: (Line 1) The BRep geometry provided by the *IfcSpace* entities is transformed into three-colored octree structures. (Lines 2-4) An inside/outside classification is performed for all octress. All octants are locally scaled. (Lines 5-8) The number of intersecting octants representing the boundaries of the spaces is computed. If the threshold value is exceeded, the two intersecting spaces are combined into a pair and added to the result list. Figure 16 illustrates the octree processing steps.

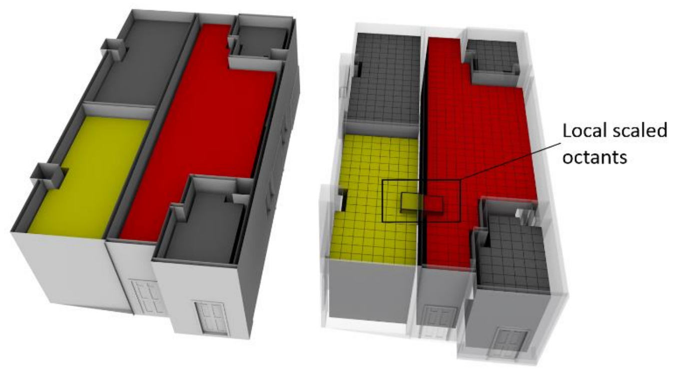


Figure 16: The left side shows two IfcSpace entities (yellow and red). The numbers of intersecting, locally scaled octants is used to verify the adjacency between the two spaces.

4.4 *Geometry based adjacency extraction*

The fourth extraction method allows to produce an adjacency graph from an IFC model even if no *IfcSpaces* are modeled. We use the third extraction method as presented before in Section 4.3. As this approach requires *IfcSpace* geometry, we automatically produce it by an octree flooding approach. The developed method is presented for one storey but can be applied for complete buildings with only minor adaptions. First, all building elements located in one storey are extracted from the model. This can be achieved by using the *IfcRelContainedInSpatialStructure* relationship or a directional-based QL4BIM query (Figure 17).

```
IfcProducts.Where(p => {  var slab1 = IfcProducts [14];
                          var slab2 = IfcProducts [15];
                          return p.Above(slab1) && p.Below(slab2); })
```

Figure 17: Extracting a single storey sub-model by QL4BIM

The received *IfcProducts* and their BRep geometry structures are stored to be used as the input for further processing. Before the process starts, the facilitated slab geometry has to be checked for breakthroughs which are removed temporally. At this point, we create an octree data structure from the BRep of the whole storey. Subsequently, the octants of the octree are colored *White*, *Gray* and *Black* by means of the inside/outside method. As we are interested in creating IfcSpace geometry which is located *inside* the storey all *White* and *Gray* octants can be deleted. Figure 18 depicted the result after the transformation of the storey's BRep into an octree structure whereas only the *Black* octants have been kept. In the next step, connected clusters are created by calling the octree's *CreateClusters* method.

By use of the volume attribute, small clusters which definitely do not represent spaces are removed. These clusters are created for example inside a column geometry. The algorithm also creates the outer hulls of the octree clusters as BRep structure. This reflects the desired shape of the *IfcSpace* geometry.

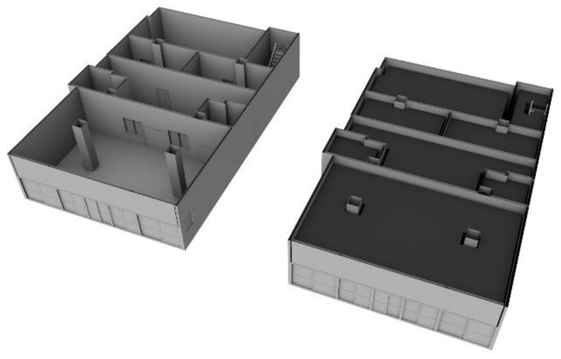


Figure 18: On the left side the connected BRep of one storey is show. The right part depicts the IfcSpaces created by the octree flooding approach.

The corresponding QL4BIM statements are shown in Figure 19. As soon as the geometry of all spaces in a storey is available, we can continue with the approach presented before in Section 4.3. The described algorithm is made available through methods of the octree data structure and can be incorporated in a QL4BIM query. The functionality of each method is explained in Section 3.3.3.

```
var brep = ProductsInStorey.ExtractOverallBRep();
var octree = brep.TransferToOctreeStucture();
octree.InsideOutsideClassification();
octree.CreateClusters();
var spaceBReps = octree.CreateOuterHull(Color.Black);
```

Figure 19: QL4BIM statements for automatic space creation

4.5 *Geometry based accessibility extraction*

Beside the adjacency of rooms also the accessibility between them is included in a Building Fingerprint. In the following, we present a method which allows to automatically deduce the accessibility graph of a

storey by QL4BIM functionality. The method uses the less possible amount of semantic information. We assume that no *IfcSpace* entities are present in the examined model. For this reason, we use the method described in Section 4.4 to automatically create spaces. Additionally, we also create octree structures for *IfcOpeningElements* by utilizing their BRep geometry. We only keep the *Gray* octants of the trees and again scale all of them locally to an extent that is greater than the maximal wall width. By testing space octants and opening octants for intersection, we receive information about which spaces are connected by which breakthrough (
Figure 20).

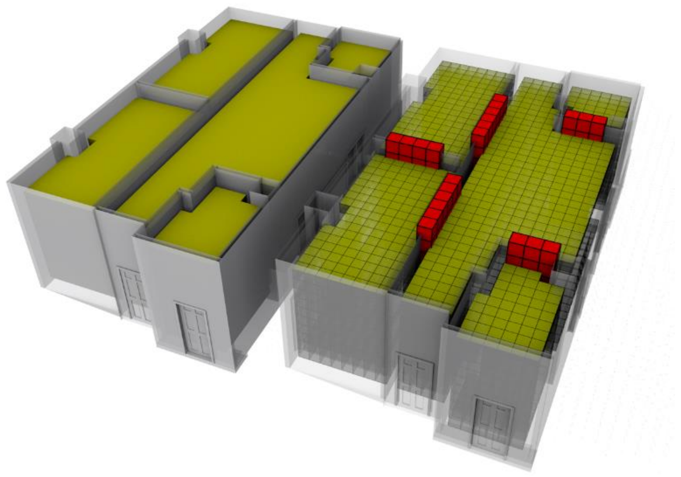


Figure 20: The left side shows IfcSpaces in one storey. The spaces and all opening elements (lifted for visualization) are transfered to octrees. Intersection counting is used to deduce accessibility between rooms (right side).

5 FROM QL4BIM RESULTS TO GRAPHS

The presented QL4BIM queries return a set of 1:1 relations between two entities. This result matches the definition of a graph G = (V, E) where V is a set of vertices and E a set of edges. Every edge relates two vertices. In the described domain, a vertex represent an *IfcSpace* for example and an edge the adjacency between two *IfcSpaces*. Based on this observation, QL4BIM is used to generate both the adjacency as well as the accessibility graph. The generated graphs are associated with the corresponding BIM as Building Fingerprint and stored in the BIM repository, where they serve as basis for indexing and querying as described in Section 2.

6 CONCLUSION

This contribution presents a new approach to automatically extract adjacency/accessibility graphs from IFC models by means of the QL4BIM query language. As the necessary semantic information is often not present in an IFC model, the presented approach focuses on creating the lacking information by investigating the available geometric data.

Combining the explained methods, adjacency / accessibility graphs can be extracted efficiently and automatically from complex IFC models. This prevent civil engineer to spend large time periods executing tedious and error prone manual work for modeling the corresponding entities and extracting adjacency/accessibility information from an IFC repository. Additionally, the presented methodology make the application of Building Fingerprints in early design stage more attractive: The efforts to set up and maintain the necessary reference repository are drastically reduced.

7 REFERENCES

Akenine-Möller, T., 2001. Fast 3D triangle-box overlap testing. Journal of Graphics Tools 6 (1), 29-33.

Daum, S., Borrmann, A., 2013a. Boundary Representation-Based Implementation of Spatial BIM Queries, in: Proc. of the EG-ICE Workshop on Intelligent Computing in Engineering, Vienna, Austria. 2013.

Daum, S., Borrmann, A., 2013b. Definition and Implementation of Temporal Operators for a 4D Query Language, in: Proc. of the ASCE International Workshop on Computing in Civil Engineering.

Daum, S., Borrmann, A., 2013c. Processing of Topological BIM Queries using Boundary Representation Based Methods: submitted. Advanced Engineering Informatics (Special Issue EG-ICE).

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

ISO 16739:2013. Industry Foundation Classes (IFC) for Data Sharing in the Construction and Facility Management Industries.

Langenhan, C., Weber, M., Liwicki, M., Petzold, F., Dengel, A., 2013. Graph-based retrieval of building information models for supporting the early design stages. Advanced Engineering Informatics 27 (4), 413–426.

Meijer, E., Beckman, B., Bierman, G., 2006. LINQ: reconciling object, relations and XML in the .NET framework, in: , Proceedings of the 2006 ACM SIGMOD international conference on Management of data. ACM, Chicago, IL, USA, pp. 706-706.

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

Weise, M., Liebich, T., Tulke, J., Bonsma, P., 2009. IFC support for model-based scheduling. CiBW78.