Thermal analysis of IFC building models using voxelized geometries

Thomas Krijnen, t.f.krijnen@tudelft.nl
Delft University of Technology, The Netherlands

Tamer El-Diraby, tamer@ecf.utoronto.ca
University of Toronto, Canada

Theohar Konomi, konomi.theohar@gmail.com
TELUS, Canada

Ahmed Attalla, ahmed.attalla@mail.utoronto.ca
University of Toronto, Canada

Abstract
Multi-disciplinary three-dimensional representations of building designs are available in BIM. However, thermal analysis often remains a manual task due to disparities in modelling: In IFC, buildings are decomposed into sets of elements with individual solid volumes. Thermal analysis requires interfaces between pairs of adjacent spaces.

In this work, we present an end-to-end solution for thermal analysis on IFC building models. We derive thin-walled thermal interfaces from architectural building models using voxelization. Neighboring spaces (separated by wall volumes in IFC) are touching after aligning to the voxel grid. The resulting mesh is converted into OSM files for OpenStudio/EnergyPlus.

This paper provides an implementation, discussion and initial validation. The algorithm depends on a minimal amount of information and favors robustness over accuracy, algorithms on voxels are robust. It is suitable for early design feedback and partial or imperfect scan2bim models.

Keywords: BIM, IFC, Geometry, Thermal analysis, Energy simulation, Renovation

1 Introduction

The Industry Foundation Classes (IFC) are a prevalent open standard to exchange Building Information Models (BIM). In such a model a geometric representation is provided for individual building elements along with user-extensible and multi-disciplinary semantic information. The geometrical definitions by which representations can be constructed are derived from existing standards for exchanging Computer Aided Design (CAD) geometries and enriched with domain specific parametric cross section profile definitions and more. Coming to a full evaluation of the building element geometries requires a considerable software implementation effort and is computationally intensive due to complex sweep operations and Boolean operations.

A typical IFC file contains individual solid volume geometries for the respective building elements. However, for other use cases, such as thermal simulation, a more topological graph-based view is required where pairs of neighboring spaces are connected by means of a single thin-walled interface. Provisions for this are incorporated in the IFC schema by means of the IfcRelSpaceBoundary (Bazjanac 2010) a subtype of IfcRelConnects. A generic framework for inferring and validating such geometric relationships is given in Krijnen et al (2020). In theory,

an IFC model can be exported with such information (Jeong et al 2014), but rarely such information is complete and the use of such Model View Definitions creates fragmentation[1].

1.1 Computational geometry

There are various geometric models that can be used underlying an implementation of the geometric resources in IFC. A semantically good match is found in the Boundary Representation (BRep) model. It consists of a hierarchy of topological entities: Solid > Shell > Face > Loop > Edge > Vertex. Of these entities, Face, Edge and Vertex have associated geometric components: an underlying Surface, Curve and Point. At every level, a tolerance factor is applied so that, for example, a Vertex point can be some small distance away from the Edge curve that is connected to the Vertex. This tolerance is necessary because floating point arithmetic on modern computer hardware is intrinsically imprecise and some solutions to geometric intersections are as well. This tolerance notion requires careful consideration when developing algorithms on BRep models.

Other implementations of IFC operate internally on triangle meshes or polyhedra. CGAL (Fabri et al. 2000) is a software library for computational geometry that offers higher precision number types that are implemented in software. In addition, by restricting operands to polyhedra, it is possible to guarantee exact constructions. Nef polyhedra (Bieri and Nef 1988) are closed under Boolean set operations, which is not the case for the BRep model described above.

BRep, polyhedra and triangle meshes all are continuous representations that define a volume based on a separation between interior and exterior. Voxelizations, on the other hand, represent geometry as arrays of values over regularized Cartesian grids. A software toolkit for voxelization of IFC building models exists under an open-source license[2]. Similar to Nef polyhedra, voxelizations are closed under Boolean set operations: a Boolean operation of two voxel inputs will always result in another valid voxel output. Note however that the voxelization process itself is destructive and cannot be easily reversed. However, voxels provide an easy and robust processing method that scales well to buildings of any size. In this case in particular, counting algorithms and recursive flood fills can be used to separate space interior and exterior, even in cases when these are not so clearly defined.

2 State of the art

An IFC model can contain thermal interfaces for analysis by means of the IfcRelSpaceBoundary type. With this data present, tools such as presented in (Yu et al. 2013) can transform the data into a format that a simulation tool can parse. Very few modelling tools generate complete and correct interfaces though. It also depends on most of the information being contained in a single authoring environment which is not always the case.

An algorithm to automatically calculate second level space boundaries after export is provided in Lilis et al. (2017) consisting of four stages: (a) parsing, (b) geometry interpretation into a BRep, (c) boundary overlap and (d) boundary projection. Detection and fixes for a set of geometric modelling issues including interfering geometries (clashes), space definition errors where space geometry does not fill the complete bounded area, surface orientation and incomplete shells is provided in Lilis et al. (2015). Another algorithm to compute second level space boundaries and a conversion to the file format of Open Studio for thermal analysis is provided in El-Diraby et al. (2017). The steps of the algorithm are different from Lilis et al. (2017). In El-Diraby et al. (2017) the solid volumes of the bounding elements are first flattened into thin-walled surfaces and then broken into segments so that a manifold composite solid of space boundaries is obtained. Both methods in this paragraph rely on a boundary representation to represent element volumes.

[1] https://blog.buildingsmart.org/blog/the-curious-case-of-the-mvd
[2] https://github.com/opensourceBIM/voxelization_toolkit

An open-source implementation of the BRep model and associated algorithms can be found in Open CASCADE[3]. IfcOpenShell[4] is a software library that implements support for the geometrical resources in IFC and is based on Open CASCADE.

The application of voxelization to BIM for geometrical interoperability is not new. In Goldstein et al. (2014) the usage of voxelization is applied to path finding, heat transfer and temperature simulation. Geological data is often represented as voxel datasets and Arroyo Ohori et al. (2018) describes the usage of such geological data in BIM infrastructure modelling. Wang et al. (2015) documents how voxelization methods are often used to correlate point clouds with BIM models as the voxelization naturally gives the point cloud a volume and uses voxels to downsample the obtained point cloud scan. Similarly, Krijnen & Beetz (2017) uses voxelization in a point cloud segmentation step to parametrize point cloud segments onto IFC building element surfaces.

Pantaleoni (2011) describes a programmable pipeline for only the voxelization of triangle meshes, but does not detail subsequent processing operations. Baumann et al. (1998) has implemented an array database system that can also be seen as a voxel retrieval system. Being primarily a database, it explicitly excludes recursion from the set of offered computation primitives. Recursion is necessary for example to fill a volume interior based on the voxelized boundary.

Very few space boundary generation solutions provide an end-to-end solution to generate a comprehensive model that can be analyzed for example in EnergyPlus (Crawley et al. 2001), one of the leading open source validated thermal analysis solutions in active development now for two decades.

3 Method and implementation

3.1 Context

We use a voxelization and computation toolkit for BIM models to derive thermal interfaces from architectural BIM models. We want to highlight several reasons to compute such boundaries on the fly rather than relying on BIM authoring tools to calculate such additional geometries on export: (a) an export-exclusive setting would require the multitude of authoring tools to implement similar additional functionality rather than focusing on their core purpose: authoring (b) providing algorithms in a separate specific purpose toolkit allows for more rapid iterations outside of the release cycle of the authoring tools (c) more specific options can be supplied in control by the thermal simulation domain expert (d) the boundary extraction can be applied to models regardless of their export settings (repurposed older models for example or models exported with different aims in mind and therefore generated using incompatible export settings) (e) most importantly though, the use of aspect models and the usage of several distinct tools in the engineering process necessitates that information on the building is distributed over various tools and model serialization, however, the generation of space boundaries during export requires that the spaces and bounding elements are contained in the same native model, which is not necessarily the case.

The algorithm developed in this paper is part of a platform called Green 2.0, a sociotechnical web-based analytics platform which integrates social interactions with 3D BIM models. The aim of Green 2.0 is to empower building occupants to take the lead in the decision-making process by studying, negotiating, and co-creating designs for their own building. This gives building occupants a strong sense of ownership, in turn encouraging the adoption of progressive actions that can save energy and enhance sustainability. Green 2.0 was applied to an exploratory case study which allowed high school students to re-design their own building in order to enhance its energy performance. Given access to a 3D model of their school on Green 2.0, students were able to visualize and study building components– including the associated cost and energy performance. Furthermore, students had the ability to select individual building elements and replace them with alternative products with pre-determined thermal performances from an

[3] https://opencascade.com/
[4] http://ifcopenshell.org/

available product library. Green 2.0 also contains a built-in energy analysis tool which enables users to calculate the thermal energy performance of the building following these changes. A discussion interface is also available to debate issues, raise questions, suggest changes, and provide assessments/knowledge about new options.

The Green 2.0 application is delivered to users as a SaaS (Software as a Service), allowing the end user to utilize the features and utilities through a modern web browser. The application core is powered by Django, backed by Postgres databases and served by uWSGI application server. Nginx is used to load balance the requests to the application servers, and serve static resources. From uploading an IFC file, to processing it and running different analysis processes, everything happens in a resource safe, asynchronous task queue powered by RabbitMQ. This modular architecture makes it easy to scale the application both horizontally and vertically. The application allows stakeholders to create design alternatives by substituting building element products and interact with comments.

3.2 Algorithm

The work in this paper builds upon previous research initiatives outlined in El-Diraby et al. (2017). Lessons learned from that earlier prototype include an identification of the following two difficulties when developing complex algorithms on BReps and Polyhedra: (a) they are detailed geometries and involve hence computational complexity and often the geometrical detail stands in the way of a clear topological definition (b) Imprecise floating-point arithmetic and tolerance values lead to the need to carefully design algorithms. On the other hand, a voxelized model is inherently topologically valid. Especially, since the initial application of this framework is intended at retrofitting an existing school building, what is most important are the relative gains of design decisions. The imprecision that stems from the voxelization is equally present in both the before and after situation and hence provides good grounds for comparison. The algorithm is outlined below:

3.2.1 Voxelization of input geometry

There are various algorithms to do voxelization (Pantaleoni 2011). In this case we use a simple triangle-box intersection where the box represents the individual voxel extent and triangles are derived from the IFC geometry as interpreted by IfcOpenShell. The voxelization toolkit used in this paper has several optimizations such as hierarchical storage, but these are not relevant for the voxel sizes used in this paper, which is deliberately coarse to automatically close gaps between spaces due to solid wall volumes. Only IfcSpace elements are included in the set of elements to be processed in this step.

3.2.2 Voxel volume filling

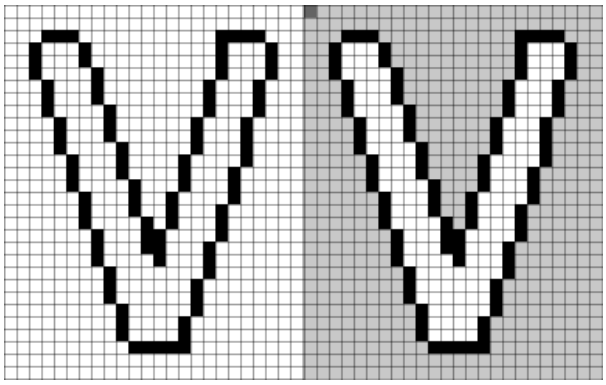

Figure 1. Procedure of finding a voxel boundary exterior. The interior is then the inversion of the exterior.

The voxelization uses triangle-box intersections, so the interior of the individual elements remains empty and unset. The thermal interface mesh generation depend on a detection of discontinuities in the grid of voxel values, where values are set to a numerical identifier derived from the IfcSpace identifiers. In order to not get discontinuities inside of the space geometries the volume inside of the voxelized boundary needs to be filled with values of the same type as the

boundary voxels. This is accomplished using recursive graph traversal where a seed point is picked known to be outside of the boundary and recursively connected neighbours are marked (see **Figure 1**). The volume is then the inversion of this marked exterior.

3.2.3 Meshing

The meshing procedure is formalized in **Figure 2**, where all voxel values are examined and compared with the three direct neighbours along the positive direction of the three axes. A set of triangles is constructed by multiplying the voxel index with the resolution. Not noted here, for simplicity, is the fact that every triangle is associated with the two space identifiers (or single space identifier in case of an element on the exterior facade).

```
1:  Ax ← {⟨1,0,0⟩, ⟨0,1,0⟩, ⟨0,0,1⟩}                    ▷ Cartesian coordinate axes
2:  procedure MESH(voxels)
3:      triangles ← ∅
4:      dim_x, dim_y, dim_z ← EXTENTS(voxels)
5:      r ← RESOLUTION(voxels)
6:      for ⟨i,j,k⟩ in {0..dim_x} × {0..dim_y} × {0..dim_z} do
7:          for Δijk in Ax do
8:              if voxels(⟨i,j,k⟩) ≠ voxels(⟨i,j,k⟩ + Δijk) then
9:                  Δa, Δb ← Ax − {Δijk}          ▷ remaining two orthogonal axes
10:                 v0 ← (⟨i,j,k⟩ + Δijk) × r
11:                 v1 ← v0 + Δa × r
12:                 v2 ← v0 + Δb × r
13:                 v3 ← v0 + (Δa + Δb) × r
14:                 triangles ← triangles ∪ {⟨v0,v1,v3⟩, ⟨v0,v3,v2⟩}
15:             end if
16:         end for
17:     end for
18:     return triangles
19: end procedure
```

Figure 2. Formalization of the meshing of a voxel grid.

3.2.4 Simplification

In our case 0.5m is used as a voxel resolution. The step above does not yield a dense mesh for visualization purposed, but for simulation purposes it creates a lot of interfaces. For that reason, there is a simplification step that takes the detailed triangle mesh as input and results in polyhedra with unnecessary intermediate vertices and edges removed. Note that, as opposed to generic mesh simplification, which involves geometrical computation and comparison to geometric tolerances, the procedure detailed below is solely topological and exact as the triangle edges are all along the grid axes, the surfaces are guaranteed planar and orthogonal.

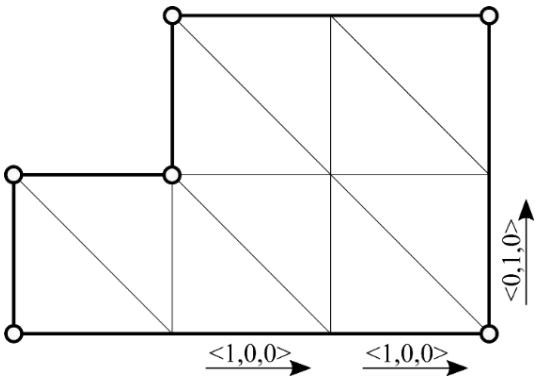

Figure 3. Overview of the simplification step. Using thin lines the origin triangulated regularized input is exemplified. In thick lines and circles one can see the simplified polyhedron derived from this.

1. All triangles are segmented by (a) their provenance with spaces (b) their surface normal and (c) the distance from the origin.
2. Border edges are isolated. These are the edges that after this segmentation are part of only a single triangle.
3. Loops along the boundary edges are found.
4. The difference vectors are computed for every edge in the boundary loop.
5. When the difference vector is different from the difference vector of the following edge in the loop, that shared vertex is part of the simplified boundary.

3.2.5 Subsurface / opening projection and property association

In this step the model is enriched with subsurfaces and thermal properties. It involves a calculation of the geometry for the openings in IFC that is matched to thermal boundary geometries based on overlap in an Axis-Aligned Bounding Box tree[5]. In cases where there is sufficient overlap detected and the orientation of the boundary is similar to the orientation of the largest face in the opening geometry, that largest face is projected onto the boundary mesh and used as a subsurface. The thermal attributes of the subsurface are derived from the properties associated to the element that fills the opening (IfcRelFillsElement).

4 Results

The algorithm is applied to a building model of a school. In **Figure 4**, **Figure 5**, **Figure 6** and **Figure 7** some of the algorithm steps and initial model geometry are displayed. There are no manual steps in the algorithm.

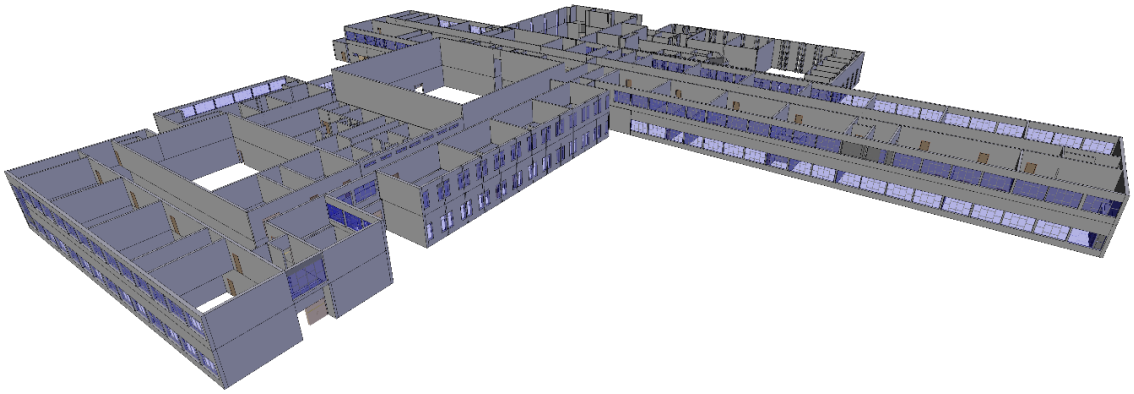

Figure 4. Overview of the building model. Only walls and openings are included to give a good overview of the model. There are some interesting non-trivial situation such as a auditorium, gym and cafeteria extending up to multiple stories.

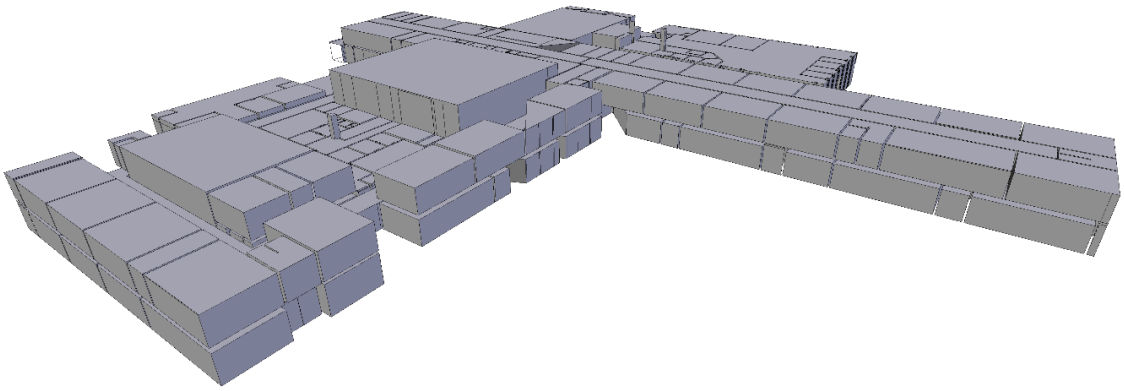

Figure 5. Geometry of the explicit spaces (IfcSpace) in the building model. It is interesting to observe there are some modelling and export errors. Some space footprints miss a boundary point and have a partly diagonal form

[5] https://github.com/kip-hart/AABBTree

as a result. Some spaces extend upwards. Both these cases result in artefacts in the final simulation model but do not cause the algorithm to terminate.

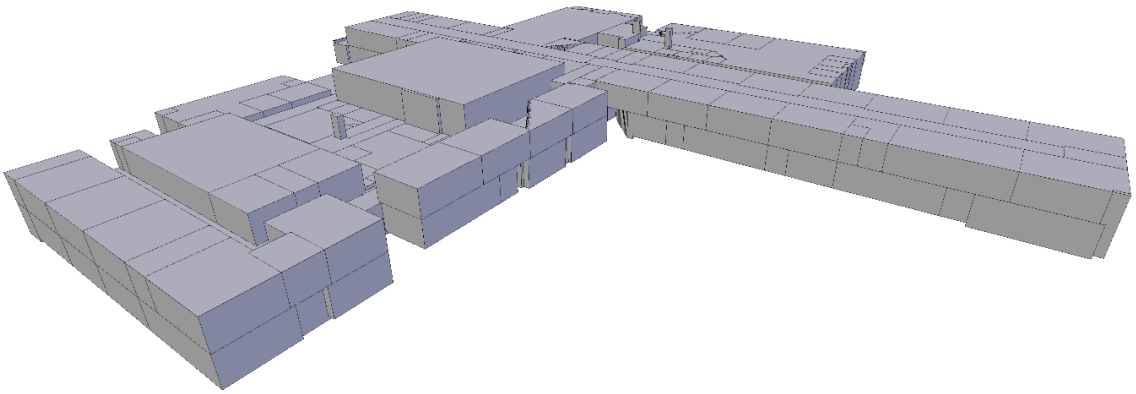

Figure 6. The simplified voxelized meshes. It's good to observe that in Figure 5 the space geometries do not touch geometrically as a result of the solid wall and slab volumes in between. In this figure the voxel resolution of 0.5m closes this distance introduced by solid-volume bounding elements.

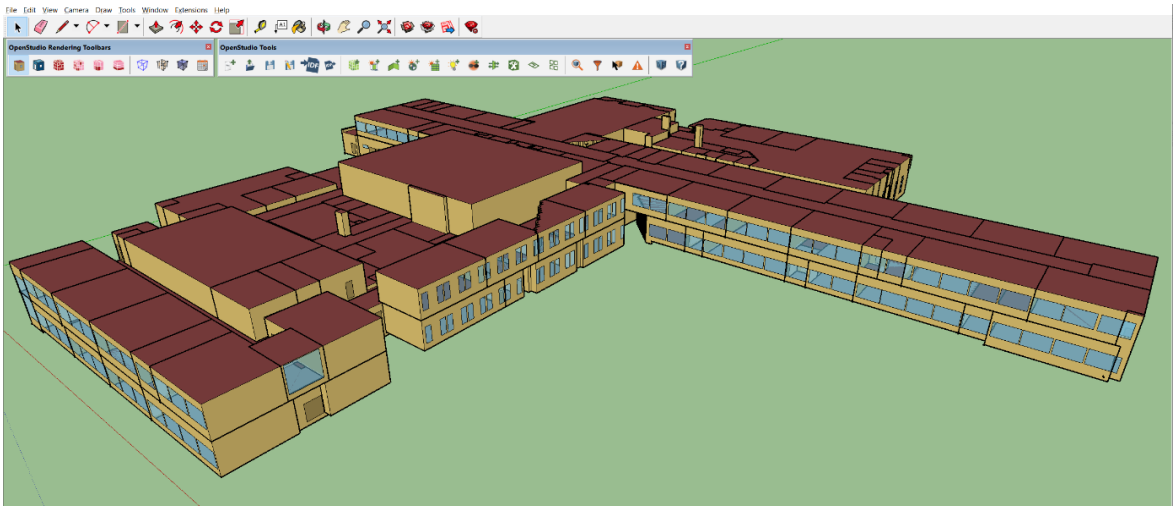

Figure 7. The resulting geometry, converted to the OpenStudio file format, with projected subsurfaces and visualized in the OpenStudio SketchUp plug-in. The generated model is valid and can serve as a valid input for a simulation in EnergyPlus.

5 Interpretation and validation

A detailed validation of the results obtained will be part of ongoing research and would include not only an assessment of the geometry, but also the topology, correctness of the associated materials and an investigation of the appropriateness of the approximations with respect to the final simulation outcomes in OpenStudio/EnergyPlus. In this paper we start with a comparison of geometric quantities in order to get an understanding of the completeness of the generated geometry.

The results in Table 1 show that the algorithm generally overestimates volumes and areas. This is to be expected the triangle-box intersection algorithm results colors a voxel even in the case of minimal overlap, therefore, on average resulting in half of the voxel size extra volume along every six directions of the unit cube. To exemplify, an isolated randomly positioned 5 × 3 × 3 box, on average would result in 5.5 × 3.5 × 3.5 dimensions on the voxel grid. This example results in a larger overestimation of volume then the voxelization outcomes in Table 1 since the space volumes in the building model are not isolated but have neighboring spaces. The one-sided thermal interface sum does match the IfcWall + IfcSlab area sum rather precisely. Important to note is that the number of cells in the thermal interface graph is less than the number of IfcSpaces

in the input model. As noted, when spaces dimensions are close to the voxel resolution the result can be that the space is not represented.

Table 1. Validation outcomes in the form of quantities derived from the original IFC model, the voxel grid and the resulting thermal interface mesh

Voxel space volume	58071.25 (464570 × 0.5^3)
IfcSpace volume sum	47257.91
One-sided thermal interface sum	36640.25
IfcWall area sum (length x height)	16683.05
IfcSlab area sum (volume / thickness)	20315.27
IfcWall + IfcSlab area sum	36998.32
Number of IfcSpaces	218
Number of unique boundary sides	208

5.1 Synthetic model

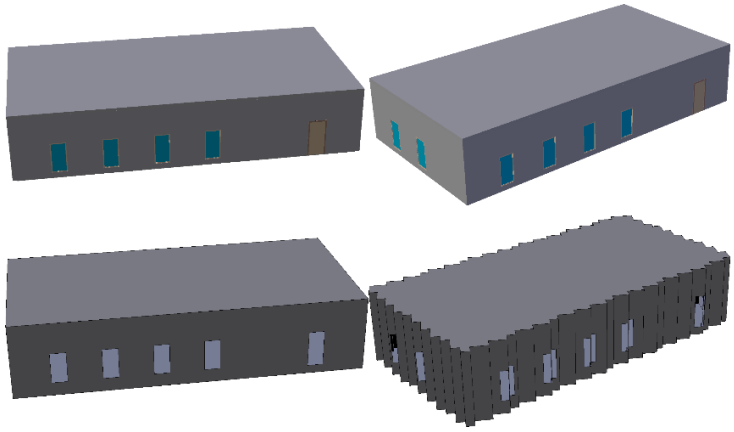

Figure 8. Graphical display of the synthetic model at two rotation angles (0° and 30°) for both input IFC model at the top and voxelized boundary mesh below.

Many building models are orthogonal and, in these cases, the voxelization grid will typically result in thermal interfaces that are slightly offset from the IFC building elements they relate to, but this difference will not result in major differences in the simulation outcomes. For non-orthogonal models or configurations where walls do not align to the axes of the voxelization grid there are more pronounced artifacts. For example, in case of a 45-degree angle, the difference in surface area for the original surface and voxelized surfaces is $\sqrt{2}$ (+41.4%). Also, the amount of boundary surfaces increases, which influences the performance of the EnergyPlus application. To illustrate the effect for this we have created a synthetic IFC model that we incrementally rotate (**Figure 8**) and present the outcomes in this section.

As can be seen in **Figure 9**, the overestimation of boundary surface area peaks at 45 degrees. It does not follow the sine function precisely because the floor and roof surfaces are not equally affected. The overall runtime of the simulation in EnergyPlus and predicted energy usage follow a similar trend. The exact inner working of the simulation steps still need to be better understood to judge the discrepancies, but self-shadowing due to the stair-case like footprint and irradiance differences that stem from the rotation are likely causes. Future investigations will be directed at seeing whether it is possible to better calibrate this solution by balancing the increase in surface area with lower transmittance coefficients.

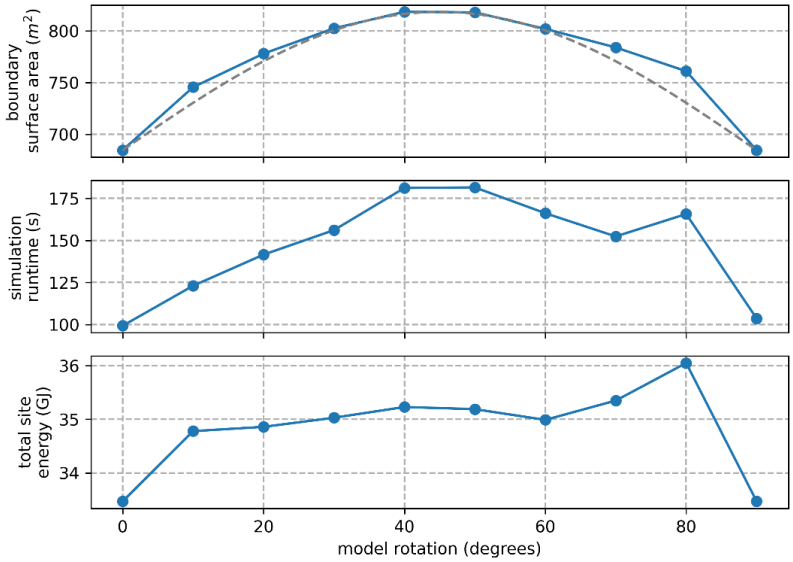

Figure 9. Parameters obtained from simulation and analysis for the incrementally rotated synthetic model.

6 Conclusion

In this paper a system to generate thermal interfaces from voxelizations of the IfcSpace geometries is presented and discussed. The solution provides an end-to-end conversion from IFC to the OSM file format ready to be used in OpenStudio/EnergyPlus. While a detailed validation and calibration of the results will be provided as part of further research, the relative ease at which the solution was developed and the fact that clear modelling and export errors (although reflected in artefacts in the final outcome) did not impact the robustness of the conversion, is promising. Discrepancies in quantities such as surface area and volume were observed and are caused by the voxelization step that discretizes the geometries over regular axis-aligned grids and the simple triangle-box intersection algorithm in use. Further research will be directed at calibrating these outcomes. As the overestimation of surface area and volume is well-understood, it will be interesting to see if this overestimation can be weighted into the thermal transmittance coefficients.

6.1 Future work

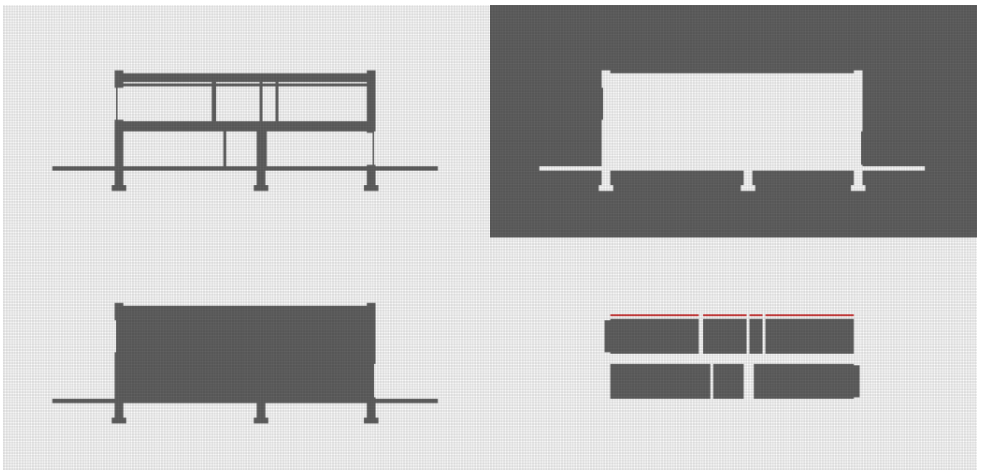

Figure 10. Illustration of the process to generate indoor spaces using voxelization: the voxelized IFC geometries (top-left); the exterior volume of the building obtained by flood fill from a point outside (top-right); the interior of the building obtained by inverting the exterior (bottom-left); the interior spaces obtained by Boolean subtraction of the interior and input geometries with potentially culling away small cavities for example from lowered ceilings by using erosion or counting (bottom-right).

The current approach outlined in this paper depends on minimal information to be available in the IFC file, but still depends on explicit IfcSpace elements to be present. Further research will be directed at using the voxelization algorithms to geometrically infer the position of spaces. **Figure 10** highlights that in several steps, starting from the voxelized geometries, a flood fill, inversion and Boolean subtraction volumes for the spaces are rather easily obtained.

In this research, the space categories are mapped to space types in EnergyPlus to realistically estimate the loads and set points, but by using machine learning, as shown in Krijnen & Tamke (2014), building elements and the functional use of spaces can be classified by which it would be possible to automatically infer design and heating constrains from the generated spaces.

References

Arroyo Ohori, K., Diakité, A., Krijnen, T., Ledoux, H., & Stoter, J. (2018). Processing BIM and GIS models in practice: experiences and recommendations from a GeoBIM project in the Netherlands. ISPRS International Journal of Geo-Information, 7(8), 311.

Baumann, P., Dehmel, A., Furtado, P., Ritsch, R., & Widmann, N. (1998). The multidimensional database system RasDaMan. In Proceedings of the 1998 ACM SIGMOD international conference on Management of data (pp. 575-577).

Bazjanac, V. (2010). Space boundary requirements for modeling of building geometry for energy and other performance simulation. In CIB W78: 27th International Conference.

Bieri, H., & Nef, W. (1988, March). Elementary set operations with d-dimensional polyhedra. In Workshop on Computational Geometry (pp. 97-112). Springer, Berlin, Heidelberg.

Crawley, D. B., Lawrie, L. K., Winkelmann, F. C., Buhl, W. F., Huang, Y. J., Pedersen, C. O., ... & Glazer, J. (2001). EnergyPlus: creating a new-generation building energy simulation program. Energy and buildings, 33(4), 319-331.

El-Diraby, T., Krijnen, T., Papagelis, M. (2017). BIM-based collaborative design and socio-technical analytics of green buildings. Automation in Construction, 82. pp. 59-74.

Fabri, A., Giezeman, G. J., Kettner, L., Schirra, S., & Schönherr, S. (2000). On the design of CGAL a computational geometry algorithms library. Software: Practice and Experience, 30(11), 1167-1202.

Goldstein, R., Breslav, S., & Khan, A. (2014). Towards voxel-based algorithms for building performance simulation. In Proceedings of the IBPSA-Canada eSim Conference.

Jeong, W., Kim, J. B., Clayton, M. J., Haberl, J. S., & Yan, W. (2014). Translating building information modeling to building energy modeling using model view definition. The Scientific World Journal, 2014.

Krijnen, T., & Beetz, J. (2017). An IFC schema extension and binary serialization format to efficiently integrate point cloud data into building models. Advanced Engineering Informatics, 33, 473-490.

Krijnen, T., Noardo, F., Ohori, K. A., Ledoux, H., & Stoter, J. (2020). Validation and Inference of Geometrical Relationships in IFC. In Proceedings of the 37th International Conference of CIB W78, Sao Paulo.

Krijnen, T., & Tamke, M. (2015). Assessing implicit knowledge in BIM models with machine learning. In Modelling Behaviour (pp. 397-406). Springer, Cham.

Lilis, G. N., Giannakis, G. I., & Rovas, D. V. (2017). Automatic generation of second-level space boundary topology from IFC geometry inputs. Automation in Construction, 76, 108-124.

Lilis, G. N., Giannakis, G. I., Rovas, D. V. (2015). Detection and semi-automatic correction of geometric inaccuracies in IFC files. In 14th International Conference of IBPSA-Building Simulation 2015, BS 2015, Conference Proceedings, pp. 2182-2189.

Pantaleoni, J. (2011, August). VoxelPipe: A programmable pipeline for 3D voxelization. In Proceedings of the Symposium on High Performance Graphics. pp. 99-106).

Wang, C., Cho, Y. K., & Kim, C. (2015). Automatic BIM component extraction from point clouds of existing buildings for sustainability applications. Automation in Construction, 56, 1-13.

Yu, N., Jiang, Y., Luo, L., Lee, S., Jallow, A., Wu, D., ... & Yen, J. (2013). Integrating BIMserver and OpenStudio for energy efficient building. In Computing in Civil Engineering (2013) (pp. 516-523).