Automated IFC-based mesh refinement and quality control for indoor airflow simulations using OpenFOAM

Anna Hochberger[1] and Veronika Elisabeth Richter[1]
[1]Institute of Energy Efficiency and Sustainable Building (E3D), RWTH Aachen University, Mathieustraße 30, 52074 Aachen, Germany
E-mail(s): anna.hochberger@rwth-aachen.de, richter@e3d.rwth-aachen.de

Abstract: Computational fluid dynamics (CFD) simulations are an essential tool for indoor airflow simulations and in the design process of energy-efficient buildings. Setting up CFD models typically demands considerable manual effort, including the modeling of geometry, defining boundary conditions, and meshing. This complex and labor-intensive process can be streamlined by automatically generating indoor CFD models from IFC-based Building Information Modeling (BIM) models. To integrate this CFD model setup in an existing tool, a plugin for the IFC-based multi-domain *bim2sim* simulation environment with the open-source CFD library *OpenFOAM* was developed, including automated meshing and refinements. As the accuracy of simulation results highly depends on the mesh quality, the mesh must satisfy specific quality criteria. Furthermore, balancing a sufficiently detailed solution provided by finer grids and reasonable runtimes is a persistent challenge in CFD. This paper presents algorithms for automatic IFC-based mesh refinement for interior elements like radiators and air terminals. These algorithms are evaluated by refinement studies and a mesh quality analysis in a case study of a single office. The proposed optimal grid refinements can be used in further research to automatically set up CFD simulations for IFC-based thermal comfort analysis.
Keywords: BIM, IFC, CFD, OpenFOAM, Meshing

Erschienen in Tagungsband 35. Forum Bauinformatik 2024, Hamburg, Deutschland, DOI: 10.15480/882.13503

1 Introduction

Computational Fluid Dynamics (CFD) simulations are a crucial tool for analyzing indoor airflow, heat distribution, and thermal comfort in buildings. Although CFD simulation results can be beneficial in the early stages of the building design process, setting up CFD models is not as straightforward as anticipated by companies within the construction industry [1]. The missing interface to open exchange formats for semantic building data is one of the existing challenges. Building Information Modeling (BIM) data can be represented using the Industry Foundation Classes (IFC), which are an open international standard for data exchange in BIM [2]. The IFC format provides a structure for storing building data, including space boundaries that can form an airtight room, which is needed for

performing CFD simulations. Unfortunately, most CFD tools, particularly those that are open-source, are not equipped to import this data format, especially its latest standard, IFC4 [1, Tab. 7.3].
To address parts of this complex problem, the open-source *bim2sim* library (https://github.com/BIM2SIM/bim2sim) was designed as a multi-domain building simulation tool that automates the simulation model setup based on IFC. For geometry processing, *bim2sim* uses the open-source geometry kernel *OpenCascade* (https://dev.opencascade.org/). The *bim2sim* tool provides plugins for multiple approaches like Building Performance and System Simulation, Life Cycle Assessments, and CFD simulations. Within the *bim2sim* tool, a CFD plugin has been developed using the open-source CFD kernel *OpenFOAM* (https://www.openfoam.com/) [3]. The goal of the *PluginOpenFOAM* is to automatically mesh, set up, and run CFD simulations for a room extracted from an IFC file. Here, automating the meshing process is one of the central challenges since the mesh quality also significantly impacts the quality of the simulation results. Therefore, this paper focuses on evaluating and enhancing the generated mesh within the plugin.

2 Related Research

Within the *bim2sim* project [1], initial investigations of IFC-based CFD-simulations have been conducted using the commercial software *ANSYS* (https://www.ansys.com/). Even though it was also implemented as a plugin for *bim2sim*, the automatic meshing and setup of boundary conditions has not been developed within this project. Despite *ANSYS* providing a variety of meshing tools, the new *PluginOpenFOAM* builds upon the *OpenFOAM*-kernel to align with the goal of creating open-source software. In this context, *OpenFOAM* also provides the advantages of being usable without a graphical user interface and its seamless integration into the codebase of *bim2sim*.
A lot of research has been conducted regarding mesh quality and improvement. Fundamentals can be found in Knupp et al. [4], who presented a mathematical derivation for a variety of mesh quality assessment criteria based on the Jacobi matrix, which have been implemented similarly in the *OpenFOAM* functionality `checkMesh`. A selection of these criteria can also be found in many commercial and open-source simulation tools (e.g. ANSYS, SimScale, MATLAB). However, in related research, no extensive investigation of full *OpenFOAM*-generated meshes has been conducted, and methods for automatically adjusting refinement levels and regions for geometric constructions like a room with furniture can hardly be found. Lee et al. [5] presented BIM-based CFD simulation setups with slightly simplified geometries. They discussed grid optimization with the focus on improving the mesh cell quality and finding an overall optimal grid size using improved grid independence tests [6]. However, their research implies that the cell quality is an indicator of overall mesh quality. It neglects the option of creating accurate meshes based on geometric and physical aspects of individual objects in the model (e.g., boundaries, air terminals, heaters, furniture), which is discussed in this paper.

3 Methodology

3.1 Object-based Meshing in OpenFOAM

The *PluginOpenFOAM* meshing process utilizes the geometry of the room including furniture combined with IFC-based semantic information, e.g., on surface types. For this, first a `blockMesh` of a specified

grid size Δx_{bM} is created, which divides the whole domain into a uniform grid. In a second step, `snappyHexMesh` refines this uniform grid using refinement levels at the object surfaces and in specified refinement regions.

One refinement level corresponds to halving the current cell width. Hence, the new grid size Δx_{new} [m] at refinement level l can be computed from the `blockMesh` size Δx_{bM} [m] as:

$$\Delta x_{new} = \frac{\Delta x_{bM}}{2^l}. \tag{1}$$

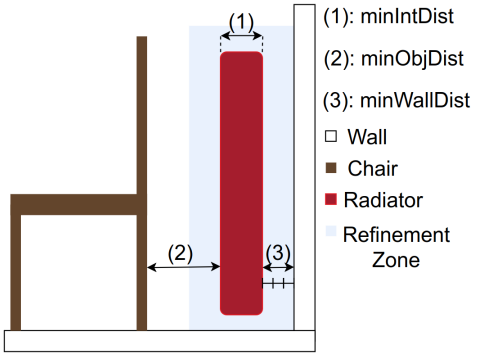


Figure 1: Relevant distances for automatic refinement.

Introducing automatic refinement is especially important closer to the walls, around objects, and at complex object geometries. For this, the three distances shown in Figure 1 become of relevance: (1) the smallest distance between two vertices inside an object (`minIntDist`), (2) the smallest non-zero meshing distance between an object and the closest room surface (`minWallDist`), and (3) the smallest meshing distance between an object and any other object inside the room (`minObjDist`). Refinement levels $refLevel$ are defined through an upper and a lower limit l, where the upper limit is chosen as $l + 2$. According to the desired final cell size, the $refLevels$ can be computed from the up-rounded Equation 1:

$$refLevel(minDist, \Delta x_{bM}) = [l, l+2], \text{ where } l = \left\lceil \frac{log(\frac{\Delta x_{bM}}{minDist})}{log(2)} \right\rceil. \tag{2}$$

3.2 Evaluating Mesh Quality

The overall quality of a mesh must be evaluated regarding the quality of the single cells and regarding its geometric correctness and completeness. For the evaluation on cell level, *OpenFOAM* provides the `checkMesh` functionality, which states the maximum value for four widely used criteria in comparison between all mesh cells. *Cell Openness* describes the ratio between the cell's volume and the volume of the smallest enclosing box (Upper Limit (UL): 1e-06). The *Aspect Ratio* is the ratio between shortest to longest edge of the cell (UL: 20). *Non-Orthogonality* relates to the angle between the vector connecting two neighboring cell centers and the vector normal to the connecting face (UL: 70). *Skewness* describes the cell's deformation compared to the ideal reference cell. This is usually normalized to a maximum of 1 but *OpenFOAM* does not perform normalization. Thus, an *OpenFOAM* user guide [7] states an UL of 8 for skewness.

A general mesh check based on correctness and completeness is not pre-implemented in *OpenFOAM* and is hard to generalize. In the proposed setup, the general mesh check focuses on identifying holes in the mesh. For this purpose, a method for validating the mesh volume is introduced. Here, the sum of all cell volumes $V_{discrete}$ is compared to the volume of the actual geometry $V_{geometric}$ in the form of the relative error: $\frac{\|V_{geometric} - V_{discrete}\|}{V_{geometric}} < tolerance.$

4 Implementation

For the volume validation, the volume $V_{geometric}$ of a closed model is computed from the provided STL files extracted from the IFC geometry using a built-in function from the Python STL package. Detected holes are fixed by adjusting the corresponding refinement zones and levels.
Regarding automatic refinement for complex geometries such as an air diffuser, the `minIntDist` is the most important distance. The vertices of an object are obtained from their geometry, e.g., a) using their `TopoDSShape`, which is the *OpenCascade* representation of an object internally used in *bim2sim*, or b) using the corresponding STL file. Based on these object vertices, the minimal distance between any two vertices within this object is identified.
Since objects in the code are only available in the triangulated form, Algorithm 1 removes the triangulation before computing the `minIntDist`. To only obtain the border vertices of the geometry, first, all edges connected to more than one triangle are removed. Second, collinear and coincident vertices are removed based on an algorithm introduced by Richter et al. [8].
The other two external distances can be computed using an in-built method from *OpenCascade* using their `TopoDSShape` representation or by converting corresponding STL files into this datatype using the `STLAPIReader`. The resulting distances, either the `minWallDist` or the `minObjDist`, must be divided by 3 to obtain the minimum distance `minExtDist`, which is further used to compute the resulting region refinement levels. This additional division ensures that there are always three cells in between two assigned boundary conditions.
For a typical diffuser, for example, Algorithm 2 determines the surface refinement level l_s based on the `minIntDist`, and the refinement level of the corresponding refinement region is set to $[0, l_s]$. For other object types, including simplified radiators, Algorithm 2 determines surface and region refinements using the `minExtDist`.

Algorithm 1: Detriangulation.

```
input: mesh; output: vertices
faces = getFacesFromMesh(mesh)
edges = getEdgesFromMesh(mesh)
outerEdges = []
for edge in edges:
    if edge not ∈ > 1 face:
        outerEdges.append(edge)
outerVertices = set([getVertices(edge) for edge
    in edges])
outerVertices = removeCoincidentVertices(
    outerVertices)
vertices = removeCollinearVertices(outerVertices)
```

Algorithm 2: Object Refinement Levels.

```
input: object, otherObjects, space, Δx_bM
output: surfaceRefinement, regionRefinement
intDist = minIntDist(object.vertices)
wallDist = minExtDist(object, space)
objDist = wallDist
for object2 in otherObjects:
    newDist = minExtDist(object, object2)
    objDist = min(objDist, newDist)
extDist = min(wallDist, objDist)
regionRefinement = refLevel(extDist, Δx_bM)
if intDist < extDist:
    surfaceRefinement = refLevel(intDist, Δx_bM)
else surfaceRefinement = regionRefinement
```

When performing CFD simulations that include turbulence and viscosity, it can be necessary to increase the refinement near walls. The wall refinement depends on the placement of the first cell at the wall within the boundary layer. The model implemented in the *OpenFOAM* plugin is based on the `omegaWallFunction`, which switches automatically between the so-called viscous and the logarithmic layer. Hence, no wall refinement is necessary from a physical perspective. However, fully omitting wall refinements results in a mesh with blunt edges, and a wall refinement level of [1, 2] was kept.

5 Results

5.1 Setup

Figure 2 shows the general setup of the use case. While this paper only discusses this specific example, the plugin generally supports creating models based on any given IFC space. Models typically include air conditioning consisting of air inlets and outlets, and radiators. The main challenges regarding the meshing of this setup lie in the complex diffuser air terminal geometry and the short distance of the radiator to the wall.

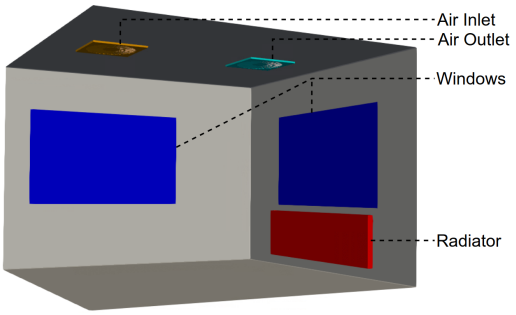
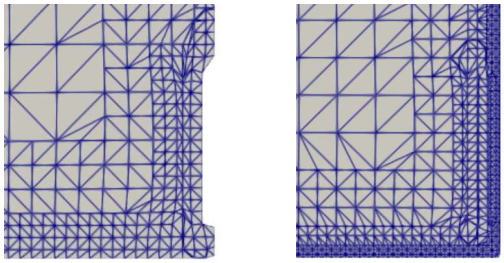


Figure 2: Room setup (Floor area: $12.95\,\mathrm{m}^2$, Height: $2.5\,\mathrm{m}$, two ext. walls).

5.2 Volume Validation

The most noticeable issue regarding correctness of the resulting mesh volume appears for meshes with $\Delta x_{bM} >= 0.14\,\mathrm{m}$ with a computed surface and region refinement level of [4, 6] for the radiator. The wall behind the heater is malformed (Figure 3a) and a deviation of $0.082\,\mathrm{m}^3$ ($0.25\,\%$ of total space volume) was computed. Even though the radiator's refinement region was computed according to Algorithm 2, fixing this hole mainly requires a wall refinement (WR) adaption of increasing the WR by one step as displayed in Figure 3b.



(a) Malformed mesh, WR = [1, 2].

(b) Fixed mesh, WR = [2, 3].

Figure 3: Meshing at radiator ($\Delta x_{bM} = 0.15\,\mathrm{m}$).

Table 1: Mesh cell quality evaluation of a set of 13 tested meshes of the use case.

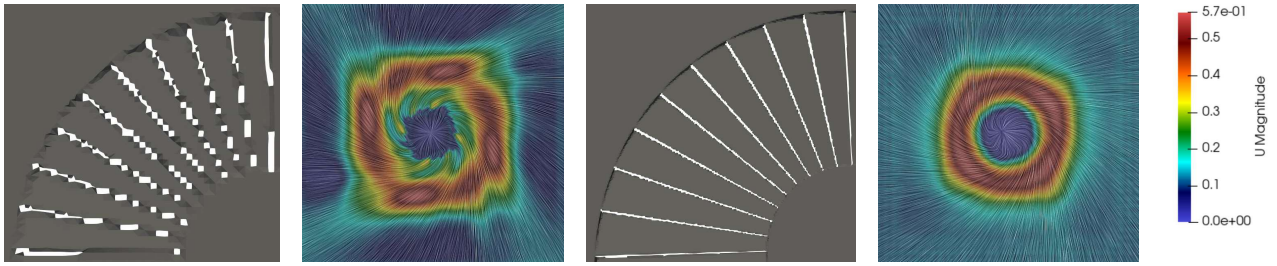
Criterion	Minimum	Maximum	Limit
Cell Openness	3.06e-16	4.29e-16	1e-06
Aspect Ratio	5.83	14.52	20
Non-Orthog.	49.55	60.00	70
Skewness	1.23	3.72	8

For other holes and gaps in the mesh, a clear difference in volume was computed and they were fixed by increasing corresponding refinement levels. Among the comparison of multiple complete meshes, the maximum deviation of $0.047\,\mathrm{m}^3$ ($0.145\,\%$) was observed for $\Delta x_{bM} = 0.1\,\mathrm{m}$ and WR = [1, 2].
Following these completeness checks, Figure 1 presents the evaluation results of the cell quality using *OpenFOAM*'s `checkMesh` functionality. These meshes range from $0.01\,\mathrm{m} \leq \Delta x_{bM} \leq 0.2\,\mathrm{m}$ including different types of air terminals, where some of these meshes contained holes. All results are well below the proposed limits and no mesh stands out as particularly good or bad. There also does not seem to be any kind of tendency, as coarser meshes did not show a worse cell quality.

5.3 Accuracy of Automatic Refinements

The level of mesh refinement of diffusers directly affects the quality of the shape representation. If the diffuser is not well refined (Figure 4b), some gaps between blades cannot be preserved and the air

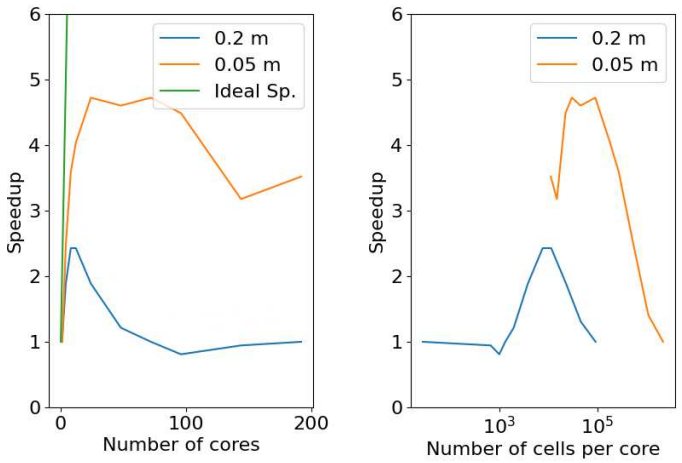
stream is disturbed. Compared to a much finer meshed diffuser (Figure 4d) the velocity distribution is less uniform. From Algorithm 1, the minimal distance in the diffuser is determined to be $1.37\,\mathrm{mm}$, leading to a refinement level of [7, 9] for $\Delta x_{bM} = 0.1\,\mathrm{m}$. For this improved refinement, differences of 0.01 - $0.05\mathrm{m\,s^{-1}}$ can be observed over the full room, with base velocities ranging from 0 - $1\mathrm{m\,s^{-1}}$.



(a) Coarse mesh ([4, 6]). (b) Coarse stream. (c) Fine mesh ([7, 9]). (d) Fine stream. (e)

Figure 4: Top view of parts of a meshed diffuser and its stream using different mesh resolutions.

5.4 Meshing Performance Evaluation



(a) Speedup over cores. (b) Speedup over cells.

Figure 5: *OpenFOAM* meshing process including mesh de- and reconstruction.

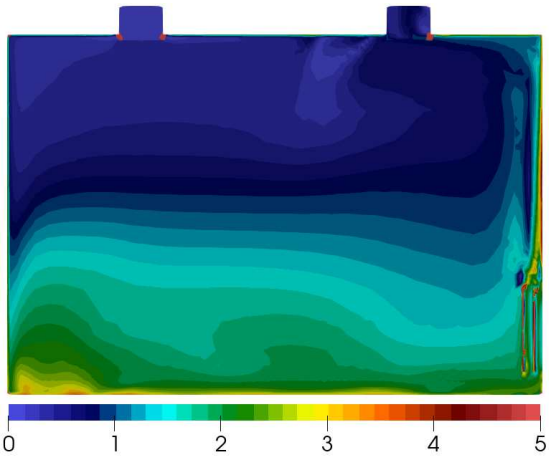


Figure 6: Root Mean-Squared Error of temperature between coarse and fine mesh (slice through air terminals (top) and radiator (right)).

To evaluate the performance of the meshing process in *OpenFOAM*, an analysis of a coarser mesh of $\Delta x_{bM} = 0.2\,\mathrm{m}$ with a total of $0.47 \cdot 10^6$ vertices and a finer mesh of $\Delta x_{bM} = 0.05\,\mathrm{m}$ with $2.6 \cdot 10^6$ vertices is presented. Figure 5a shows the speedup from 1 to 192 cores on the RWTH Computing Cluster CLAIX-23 for both meshes. It is noticeable, that the speedup only increases for a very small number of cores in both cases. Even though the speedup for the coarser mesh is a lot lower ($17\,\mathrm{s}$ for 1 core) than the one of the finer mesh ($359\,\mathrm{s}$ for 1 core), both stagnate or decrease for more than 24 cores. Figure 5b shows that the speedup peak not only depends on the number of cells per core but also on the total mesh size due to increased communication for a higher number of cells.
Figure 6 presents the temperature error between the two meshes, referring to an absolute temperature range from 285 - $310\,\mathrm{K}$ calculated as a baseline using the fine mesh. Differences of 1 - $3\,\mathrm{K}$ are visible in huge parts of the room. Around and above the radiator, the differences are even higher. For the air

velocity, ranging from 0 - $1.02\,\mathrm{m\,s^{-1}}$, differences of 0 - $0.15\,\mathrm{m\,s^{-1}}$ are distributed over the full room. Additionally, an investigation of the Grid Convergence Index (GCI) could have been very informative as a verification that the results converge to the analytical solution with decreasing grid size. Unfortunately, the GCI sets relatively high demands on the grids, especially regarding the number of vertices, which could not be fulfilled in this setup. However, the evaluation of the GCI might become more feasible when investigating larger room models.

6 Discussion & Limitations

Overall, the cell quality of all meshes is in a good range and does not seem to be a potential source of errors for the evaluated setups. The selection of criteria that the `checkMesh` functionality provides includes the most important evaluation criteria, which could also be provided by other tools. However, the observed well-formedness of the cells could change as soon as furniture or other objects with a more complex geometry are included in the CFD model. While the algorithmic validation of the completeness of the mesh gave very clear results, two problems remain. First of all, the limit for the deviation of the volumes is not set yet. Even for complete meshes, the deviation is not constant and it remains unclear, how well only a few missing cells could be detected by this algorithm. Secondly, it is important to identify the location of holes or missing cells automatically, to be able to specify the region that needs to be refined. As of now, fixing holes therefore remains a mainly manual process that requires more extensive research.

Regarding the accuracy of the mesh refinements, the proposed algorithms provide an effective framework for automatically detecting relevant refinement sizes. Since this framework can be based on STL files, it can also be applied outside the *bim2sim* environment. Figure 4 shows that the meshing on individual elements can have an impact on the full room and therefore proved the importance of geometrically accurate meshing. However, the holes in Figure 3a showed that it might be necessary to extend Algorithm 2 to also adapt the wall refinements in the future.

Independent of determining the desired refinement levels, meshes often did not turn out as expected. Even though a diffuser surface refinement level of [6, 8] gave great results for $\Delta x_{bM} = 0.1\,\mathrm{m}$, these results could not be reproduced for finer meshes, especially with $\Delta x_{bM} < 0.08\,\mathrm{m}$. For both, higher and lower refinement levels and variations in the refinement region size, the diffuser came out very coarsely meshed. The reasons for this are unresolved since there was no observable pattern throughout a variety of tests and this issue continues to exist for further research. Additionally, other meshing tools (e.g., SALOME, CFMesh) should be evaluated regarding a more reliable mesh quality.

The meshing performance evaluation highlighted that parallelization is not very efficient for meshing, mostly due to the high overhead from distributing the relatively small room onto many processes. It may therefore be useful to check the runtimes again when performing simulations of larger rooms. Generally, the runtimes of the meshing are relatively small compared to the actual simulation times and should not be given too much weight when making parallelization decisions. The mean-squared error proved that the mesh size has a significant influence on overall simulation results. While it is usually the case that finer meshes provide more accurate solutions, this still needs verification in the future, e.g., using the GCI or grid independence tests as adapted by Lee et al. [6].

7 Conclusion

This paper presented a basis for setting up automatic object refinements and evaluating IFC-based meshes using *OpenFOAM* within the *bim2sim* simulation environment. Besides classic cell quality criteria, a method for checking the completeness of a mesh was introduced. Automatic refinement was introduced for surface and region refinements based on the minimal internal distance of a detriangulized object and the minimal distance to any other surface in the room. Though these algorithms form a widely usable framework for automatically improving meshes in the context of BIM, they still require more extensive testing to guarantee stability which are subject of further research.

Acknowledgements

This research was funded by the Federal Ministry for Economic Affairs and Climate Action within the project “BIM2Praxis” (grant number 3EN1050A). Simulations were performed with computing resources granted by RWTH Aachen University under project thes1667.

References

[1] D. Müller, D. P. Jansen, C. A. van Treeck, *et al.*, *BIM2SIM - Methodenentwicklung zur Erstellung von Simulationsmodellen aus Daten des Building Information Modeling : gemeinsamer Endbericht*. 2021. DOI: 10.2314/KXP:1819319997.

[2] ISO 16739-1, *Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries - Part 1: Data schema.* 2018.

[3] V. Richter, C. van Treeck, and J. Frisch, “Extending an ifc-based framework to include an automated cfd-setup using pre-computed boundary conditions”, in *Proceedings of BauSIM Conference 2024*, ser. BauSIM Conference, Wien, Austria: IBPSA DACH, 2024.

[4] P. M. Knupp, “Algebraic mesh quality metrics”, *SIAM Journal on Scientific Computing*, vol. 23, no. 1, pp. 193–218, 2001. DOI: 10.1137/S1064827500371499.

[5] M. Lee, G. Park, H. Jang, and C. Kim, “Development of building cfd model design process based on bim”, *Applied Sciences*, vol. 11, no. 3, p. 1252, Jan. 2021. DOI: 10.3390/app11031252.

[6] M. Lee, G. Park, C. Park, and C. Kim, “Improvement of grid independence test for computational fluid dynamics model of building based on grid resolution”, *Advances in Civil Engineering*, vol. 2020, W. Tangchirapat, editor, pp. 1–11, Dec. 2020. DOI: 10.1155/2020/8827936.

[7] WolfDynamics, *Advanced meshing using openfoam technology: Cfmesh*.

[8] V. Richter, A. Malhotra, E. Fichter, A. Hochberger, J. Frisch, and C. van Treeck, “Validation of ifc-based geometric input for building energy performance simulation”, in *Proceedings of the 2022 Building Performance Analysis Conference and SimBuild*, ser. ASHRAE/IBPSA-USA Building Simulation Conference, ASHRAE and IBPSA-USA, 2022. DOI: 10.26868/25746308.2022.C033.