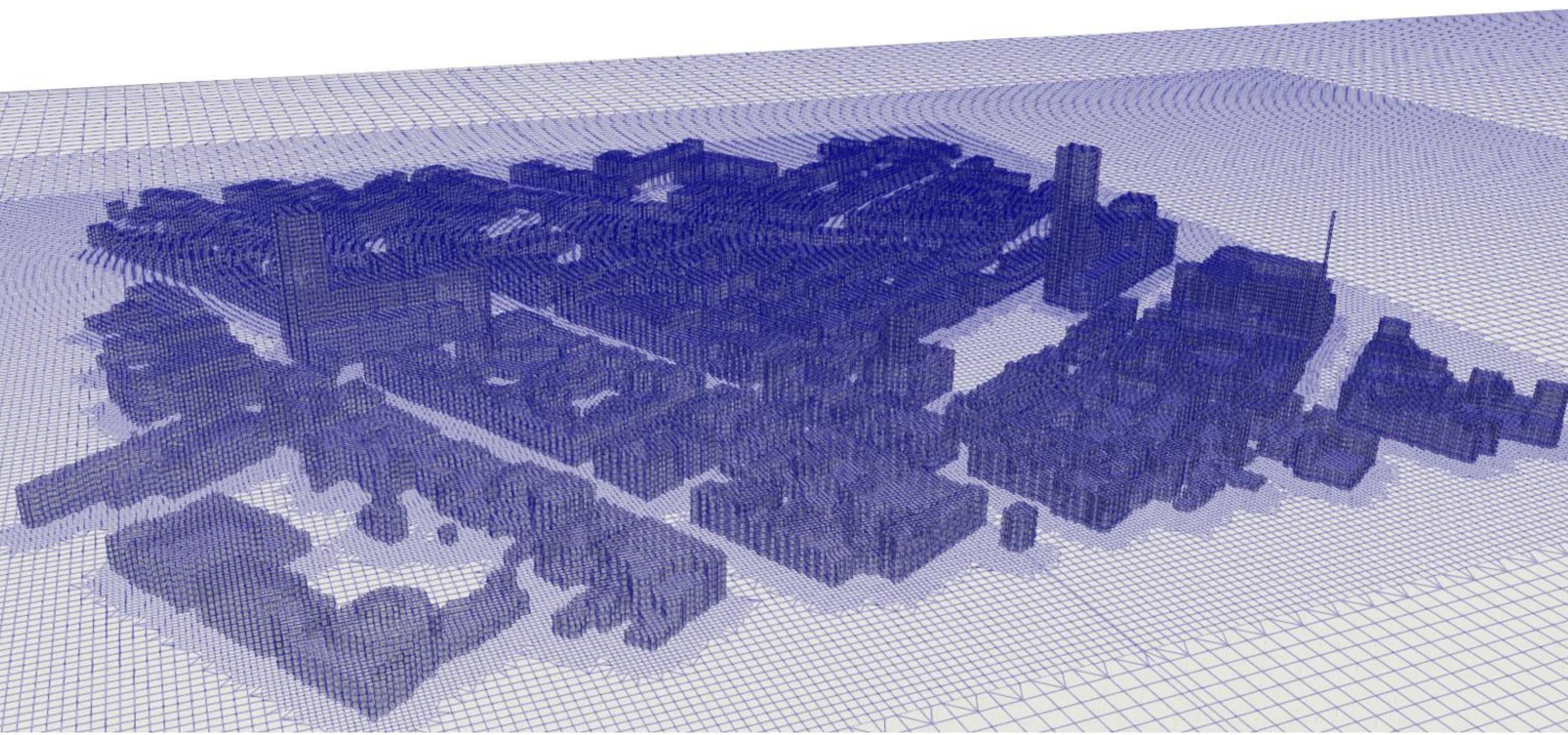
Urban computational fluid dynamics simulations set-up validations

Master thesis in Geomatics

Maren Hengelmolen
January, 2024



MSc thesis in Geomatics

Urban computational fluid dynamics simulations set-up validations

Maren Hengelmolen

January 2024

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Clara García-Sánchez
	Hugo Ledoux
Co-reader:	Azarakhsh Rafiee

Abstract

Computational Fluid Dynamics (CFD) simulations for residential areas are becoming increasingly important as the urban population keeps growing and extreme weather conditions due to climate change are becoming more prevalent. Since CFD simulations simulate and visualise fluid flows, users can analyse and predict air flows in urban environments, giving insight in pollution, heat dissipation, and weather conditions. Performing these simulations demands expertise and time: 3D urban models must be prepared, and pre-run setups must be defined to obtain accurate results. Partial automation can streamline this process. Therefore, we developed a method that identifies geometric errors and defines mesh parameters for the open source CFD software OpenFOAM. This method follows the ISO19107 standard and recent CFD guidelines for urban areas, ensuring accurate simulation results. We validated algorithms with a variety of urban models and parameters, and analysed the workflow developed for the mesh parameters definition. Additionally, we created a prototype, in the form of a web application, in which these algorithms are implemented. Our prototype will simplify the use of CFD simulations for urban areas, making them more accessible to everyone.

Acknowledgements

This thesis is written as a master end project for the Master Geomatics at Delft University of Technology. My interest in three dimensional data and CFD simulations led me to this topic. In this thesis, I explore ways to simplify the use of urban CFD simulation. I developed a prototype, using C++ and Python programming, that handles some preparation tasks of geometries and pre-run setups for CFD simulations. These tasks are time-consuming , prone to errors and require some expertise. Using the prototype discussed in this thesis, users get feedback on their model and pre-run setups that are required for realistic simulation results.

I would like to thank my supervisors Clara García-Sánchez and Hugo Ledoux, for their time and effort in guiding me. They were always ready to help me out, even though their schedules were filled to the brim with appointments. I would also like to thank my co-reader, Azarakhsh Rafiee, for her valuable feedback on the report, and Peter van Oosterom for his interest and his thought-provoking questions during the official meetings.

A big thank you to my friends for the support and necessary distraction. I would like to express my deepest gratitude to my parents and sister Vera for their support and encouragements. Last but not least, I would like to thank Joep, who had to hear all my complaints and doubts, for his support, patience and advice.

Thank you all !

Maren Hengelmolen
Delft, 2024

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AIJ	Architectural Institue of Japan
BAG	Register of Buildings and Addresses
CFD	Computational Fluid Dynamics
CGAL	Computational Geometry Algorithms Library
COST	Cooperation in Science and Technology
FOAM	Open Field Operation and Manipulation
ISO	International Organization for Standardization
LES	Large Eddy Simulation
OBJ	Wavefront Object
OGC	Open Geospatial Consortium
RANS	Reynolds-Averaged Navier-Stokes
RoI	Region of Interest
Simple	Semi-Implicit Method for Pressure-Linked Equations
STL	Stereolithography
UI	User Interface

Symbols

Topological relationships validations

ζ	Distance threshold between terrain and buildings
z	Building height
z_g	Ground level
z_{min}	Minimum z-value in the 3D model
z_t	Target height or height of the terrain surfaces

Sharp angles identification

θ	Angle threshold between two triangular building faces (°)
ϕ_1	First angle between two triangular faces (°)
ϕ_2	Second angle between two triangular faces (°)

Sliver triangles identification

SP	Sliver parameter
σ	Sliver parameter threshold used to define sliver triangles

Short edges identification

l	Edge length
λ	Length threshold used to define short edges

Mesh parameters definition

BR	Blockage ratio	κ	Turbulent kinetic energy
BR_H	BR in the lateral vertical direction	N	Number of cells in mesh
BR_L	BR in the lateral horizontal direction	N_{boxes}	Number of refinement boxes
b_{target}	Target building (x, y)	$N_{approx.}$	Number of cells approximation in mesh
D_{box}	Refinement boxes dimensions (m)	$N_{blockMesh}$	Number of cells in background mesh
d_{flow}	Flow direction (°)	N_{max}	Maximum number of cells
d_{max}	Largest building dimension (m)	r_{min}	Target cell ratio
$d_{separation}$	Maximum street width (m)	u	Unit parameter
ϵ	Passive scalar	U_{ref}	Velocity value (m/s)
h_{cell}	Cell height (m)	w_{cell}	Cell width (m)
h_{max}	Height of the tallest building (m)	z_{ref}	Velocity height (m)
h_{max}	Maximum cell height (m)	z_0	Maximum roughness height (m)
h_{min}	Minimum cell height (m)		
h_{user}	Target evaluation height (m)		

1. Introduction

1.1. Background and motivation

Computational Fluid Dynamics (CFD) simulations are a powerful tool to address societal challenges within urban environments (Blocken, 2015). As they simulate and visualize fluid flows, users can analyse and predict flow behaviours within cities, such as wind, pollution, and heat flows. With the growing population in urban areas (The World Bank, 2022) and the rise of extreme weather conditions due to climate change, these CFD simulations are becoming increasingly important. This development, for example, lead to more natural ventilation and storm resistance in the build environment. These simulations are able to support urban planning, e.g. by preventing strong wind flows due to the architecture and/or distribution of buildings. In other words, CFD simulations help to create or maintain a balance between urban flows ensuring a comfortable environment within residential areas.

CFD simulations represent the air surrounding the urban area by a computational domain, which is characterised by turbulent flows displaying chaotic behaviours and described by Navier-Stokes (NS) equations. Due to their infinite degrees of freedom, these flows are very challenging to solve and ”often considered as the ”last grand challenge” of classical physics” (Benzi and Toschi, 2023). Therefore, a mesh is generated that divides the domain into cells. For each of these cells, the flow value is computed by solving a simplified version of the NS equations (Blocken, 2015; Franke and Baklanov, 2007). The quality of these meshes, which is mainly dependent on the grid resolution and cell quality, is essential for accurate simulation results (Blocken, 2015).

CFD simulations require accurate pre-run setups. These are proposed in different CFD guidelines. Properly following these guidelines ensures realistic simulation results. Currently, users must manually define these pre-run setups, which is time consuming, prone to errors, and requires some expertise. However, partial automation can streamline this process. This would help increase the ease of use of CFD simulations. The guidelines include recommendations for the meshing process, which include domain dimensions, cell sizes and cell shapes. A tool that computes these mesh parameters based on these suggestions might contribute to further popularisation of CFD simulations.

Besides this, the quality of the geometries within 3D urban models is also crucial in performing accurate CFD simulations (Wagner et al., 2015). To guide the increasing use of geometric data for spatial analyses, international standards for geographical information were developed. Standards that help create valid 3D geometric datasets and encourage the interoperability and exchange of geographical data. Though these international standards do take into account most geometric aberrations, some additional geometries can also affect the mesh quality, such as sharp angles, short edges, and sliver triangles (Pađen et al., 2022; Piepereit et al., 2018). Additionally, incorrect topological relations between buildings and terrains can lead to unrealistic simulation results. In the real world, most buildings do not have strong flows passing beneath them.

In this thesis a method is developed that defines mesh parameters and validates geometries for accurate CFD simulations in urban areas. Recent CFD guidelines and international standards for geographic information are implemented. An additional prototype in the form of a web application is also developed that executes this method. Further development of this prototype can contribute to simplifying the use of CFD simulations in urban areas.

1.2. Objective and research questions

The aim of this thesis is to develop a method that validates geometries and defines mesh parameters to simplify the use of CFD simulations in urban areas. Therefore, the most recent CFD guidelines and international standards for geographic information are used. Additionally, a prototype is created, in the form of a web application, that executes this method and could be converted to an open source tool. To achieve these objectives, the following sub-questions are addressed:

- Which geometric validations should be performed for CFD simulations in urban areas? And how to implement them?
- Which method(s) could be used to compute CFD mesh parameters and generate high quality meshes for urban CFD simulations in OpenFOAM?
- How to report errors, warnings and results to the users?
- How to validate the quality of the methodology?

1.3. Scope

The focus lies on identifying and reporting geometric errors; hence, repairing tasks are not included. The method is developed based on building and terrain features and is not tested with vegetation and other type of infrastructure. In addition, semantic validation functions are not integrated. The mesh parameters definition algorithms create configuration files for OpenFOAM, which is a widely used open-source software, to encourage the use of CFD simulations. This thesis focuses on rectangular domains since they are common (Mirzaei and Carmeliet, 2013) in CFD simulations for urban areas.

1.4. Obtained results

A prototype was created in the form of a web application to help users prepare their 3D urban models for CFD simulations. First, geometric errors and warnings are identified in accordance with ISO19107, an international standard for geographical information related to vector geometries and topology. Second, CFD mesh pre-run setups are defined based on recent CFD guidelines for urban areas. These parameters are generated in output files compatible with OpenFOAM. Finally, this prototype warns users when certain CFD guidelines cannot be satisfied.

1.5. Thesis outline

This report is organized as follows:

- Chapter 2 addresses theoretical background based on literature. It introduces CFD simulations in urban areas and discusses how 3D geometries can be validated.
- Chapter 3 describes the method used to validate 3D geometries and define CFD parameters for the meshing process. It also explains most of the developed algorithms.
- Chapter 4 provides information on the method implementation, and introduces the user interface/web application.
- Chapter 5 presents and analyses the obtained results.
- In Chapter 6, the conclusions and recommendations are drawn.

2. Theoretical background and related work

In this chapter, theory and work relevant for the development of the method, identifying geometric errors and mesh parameters definition, are presented. First, the chapter addresses CFD simulations in urban areas (Section 2.1), and relevant guidelines for accurate simulation results (Section 2.2). Subsequently, the chapter provides necessary information on geometric validations (Section 2.3), and geometric issues related to the CFD meshing process (Section 2.4).

2.1. CFD simulations in urban areas

2.1.1. Urban physics and CFD simulations

Most analyses in urban physics are conducted within the lower part of the Atmospheric Boundary Layer (ABL) (Blocken, 2015), also known as Planetary Boundary Layer (PBL) (AMS, nd). The ABL is the lowest part of the troposphere, which extends 10-20km above the ground (AMS, nd), and is directly influenced by the surface of the Earth (Brancher et al., 2017). Its height varies depending on surface properties (e.g. frictional drag, pollutant emissions, and heat transfer) (Seetha et al., 2023), and can range from several tens of meters in situations with strong statically stable situations (laminar flows) to kilometres in statically unstable situations (turbulent flows) (AMS, nd). Within the ABL, various weather phenomena can occur that can be classified on their horizontal spatial and temporal scales (Blocken, 2015; Habby, nd; AMS, nd), as shown in Table 1.

Table 2.1.: Meteorological scales.

Meteorological scales	Horizontal size	Duration	Examples of atmospheric phenomena
Macroscale	Many hundreds of kilometers	Days	Cyclones, hurricanes
Mesoscale	Several hundred kilometres	1-24 hours	Thunderstorms, precipitation bands, mountain waves, sea and land breezes
Microscale	2km or less	Minutes to hours	Tornadoes, rainbows, convective updrafts and downdrafts

In urban physics, flows that are analysed and modelled on the meteorological microscale are usually referred to as Computational Fluid Dynamics (CFD) models (Blocken, 2015). This microscale environment defines the ABL as the computational domain. Therefore, at its boundaries, physical properties such as wind velocity and direction, temperature, humidity, and pressure (Christou, 1998), are specified. These boundary conditions can be data from the meteorological mesoscale. CFD models can also be applied on a building scale, covering an even smaller area (100 metres or less) (Blocken, 2015).

2.1.2. Governing equations in CFD simulations

The flows within the ABL are turbulent (Blocken, 2015) characterized by chaotic behavior. Chaotic flows do not have a widely recognised definition, but they are often described as flows without constant or periodic motion (Liou, 2008). They can also be described as ”rotational, intermittent, highly disordered, diffusive and dissipative” (Markatos, 1986). Due to their large degrees of freedom, understanding these flows is extremely challenging and is often considered the ”last grand challenge” of classical physics (Benzi and Toschi, 2023).

In CFD, the Navier-Stokes (NS) equations are used to describe fluid motions, covering the conservation of mass, momentum, mass, energy and scalar. They can be written as follows (Blocken, 2015):

- Conservation of mass (2.1) and momentum (2.2 and 2.3)

$$\frac{\delta u_i}{\delta x_i} = 0 \tag{2.1}$$

$$\frac{\delta u_i}{\delta t} + u_j \frac{\delta u_i}{\delta x_j} = -\frac{1}{\rho}\frac{\delta p}{\delta x_i} + \frac{\delta}{\delta x_j}(2\nu s_{ij}) \tag{2.2}$$

$$s_{ij} = \frac{1}{2}(\frac{\delta u_i}{\delta x_j} + \frac{\delta u_j}{\delta x_i}) \tag{2.3}$$

  where
  - u_i: instantaneous velocity
  - x_i: instantaneous position
  - p: instantaneous pressure
  - t: time
  - ρ: density
  - ν: molecular kinematic viscosity
  - s_{ij}: strain-rate tensor

- Conservation of scalar

$$\frac{\delta \theta}{\delta t} + u_j \frac{\delta \theta}{\delta x_j} = \frac{1}{\rho c_p}\frac{\delta}{\delta x_j}(k\frac{\delta \theta}{\delta x_j}) \tag{2.4}$$

  where

  - θ: instantaneous temperature
  - k: thermal conductivity
  - c_p: specific heat capacity
- Conservation of scalar

$$\frac{\delta c}{\delta t} + u_j \frac{\delta c}{\delta x_j} = \frac{\delta}{\delta x_j}(D \frac{\delta c}{\delta x_j}) \tag{2.5}$$

  where

  - c: instantaneous concentration
  - D: molecular diffusion coefficient or molecular diffusivity

Since solving these equations is highly challenging and almost impossible due to limited computational resources, the equations must be simplified. There are two main simplification methods: Reynolds-Averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) (Blocken, 2015).

RANS is the most common simplification method and can be divided into steady and unsteady RANS (Blocken, 2015). Steady RANS is developed for steady flows, meaning that the flow conditions can vary at different locations, but not with time. The flow conditions in unsteady RANS, on the other hand, can change with both time and location (CREST Foundation Studies, nd). The RANS equations are a decomposition of the NS equations into a mean and a fluctuation component (Blocken, 2015).

- RANS equations for the conservation of mass (2.6) and momentum (2.7 and 2.8)

$$\frac{\delta U_i}{\delta x_i} = 0 \tag{2.6}$$

$$\frac{\delta U_i}{\delta t} + U_j \frac{\delta U_i}{\delta x_j} = -\frac{1}{\rho}\frac{\delta P}{\delta x_i} + \frac{\delta}{\delta x_j}(2\nu S_{ij} - \overline{u'_j u'_i}) \tag{2.7}$$

$$S_{ij} = \frac{1}{2}(\frac{\delta U_i}{\delta x_j} + \frac{\delta U_j}{\delta x_i}) \tag{2.8}$$

  where

  - U_i: mean velocity
  - P: mean pressure
  - u'_i: Reynolds stresses, which is a fluctuation component and represent the influence of turbulence on the mean flow
  - S_{ij}: mean strain-rate tensor
- RANS equations for the conservation of energy

$$\frac{\delta \Theta}{\delta t} + U_j \frac{\delta \Theta}{\delta x_j} = \frac{1}{\rho c_p}\frac{\delta}{\delta x_j}(k\frac{\delta \Theta}{\delta x_j} - \overline{u'_j \theta'}) \tag{2.9}$$

  where

  - Θ: mean temperature
  - θ': turbulent heat fluxes, which is a fluctuation component and represent the influence of turbulence on the heat transfer

- RANS equations for the conservation of scalar

$$\frac{\delta C}{\delta t} + U_j \frac{\delta C}{\delta x_j} = \frac{\delta}{\delta x_j}(D \frac{\delta C}{\delta x_j} - \overline{u'_j c'}) \tag{2.10}$$

  where

  - C: mean concentration
  - c': turbulent mass fluxes, which is a fluctuation component and represent the influence of turbulence on the mass transfer

Steady RANS uses time-average and unsteady RANS applies ensemble-average of the NS equations. Both methods do not directly simulate turbulence but rely on statistical modeling. This is because the number of unknown variables exceeds the number of RANS equations due to the introduction of new variables. Consequently, these equations cannot be solved. Turbulence models are then used to approximate Reynolds stress values. For temperatures and concentrations, the Boussinesq approximation for buoyancy is often used (Blocken, 2015).

Since the ABL is turbulent and turbulent flows are considered unsteady, using unsteady RANS is theoretically more appropriate (Blocken, 2015; COST, 2023). However, unsteady RANS requires high spatial resolution (Blocken, 2015; COST, 2023). Therefore, Franke et al. recommend using LES or hybrid URANS/LES instead. With LES, large-scale turbulence can be resolved by applying spatial filtering to the NS equations, which removes small turbulent eddies. The filter usually has the same dimensions as a grid cell at that specific location. Small-scale turbulent flows are approximated by using statistics (Blocken, 2015; COST, 2023). Hybrid URANS/LES combines URANS near walls, as LES is computationally expensive, with LES in the remaining area (Blocken, 2015; COST, 2023). Both LES and hybrid URANS/LES outperform RANS and URANS (Blocken, 2015; COST, 2023).

Despite the recommendation for unsteady methods in ABL simulations, steady RANS remains the most widely used in urban physics (Blocken, 2015; COST, 2023), and often provides satisfactory results (Blocken, 2015; Blocken, 2014). On the other hand, LES is the preferred approach for unsteady flows and is increasingly used (COST, 2023). However, the execution time is often high, and high-quality experimental data are needed as input data, which are sometimes difficult to obtain (COST, 2023). Moreover, more information and guidelines are available for RANS simulations, making it a more attractive option in many cases (Blocken, 2015). This thesis focuses on RANS simulations, because of its popularity in urban physics and the extent of information.

2.1.3. OpenFOAM, an open CFD software

To increase the ease of use of CFD simulations in urban areas, this research focuses on the widely used open source CFD software OpenFOAM. Three main steps can be identified while performing CFD simulations with this software. First, a computational domain is generated around the input geometry. Next, a mesh or computational grid subdivides this domain into cells (e.g. hexahedra, tetrahedra, polyhedra). Finally, the flow values are computed at the centre point of each cell by using one of the available solvers. These can be chosen by the users based on the fluid properties and specific simulation requirements. For each of these steps, the users must execute one or several commands/utilities (Greenshields, 2023).

The two first commands are essential to generate an adequate mesh. The first one is the *surfaceFeatures* command, which identifies the surfaces features (edges and vertices) of the input geometry. This utility enables the final mesh to properly mesh the geometry edges. Then, a computational domain is generated and divided into hexahedral cells by using the *blockMesh* utility. Its specifications, such as domain size and cell dimensions, can be added in the *blockMeshDict* file. The resulting domain is named the background mesh. It is important to note that its cells must maintain an aspect ratio of approximately 1 to aim for the highest mesh quality with *snappyHexMesh*. In addition, there must be at least one intersection between a cell edge and the input geometry. Figure 2.1 shows an example of a background mesh in 2D.

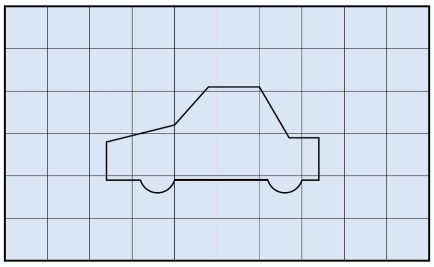


Figure 2.1.: Background mesh in OpenFOAM. Adapted from "User Guide version 11" by Greenshields, C. J., 2023.

After the background mesh generation, the final mesh can be modeled using the *snappyHexMesh* command, which is only compatible with triangulated surface geometries in Stereolithography (STL) and Wavefront Object (OBJ) files. For this reason, this research focuses on these two formats. Figure 2.2 illustrates the different steps during this meshing process. First, the cells are divided into smaller cells at the surfaces and edges of the input geometry. Second, cells with 50% or more of their volume within the geometry are removed. Third, cells located within user defined regions are refined. Finally, vertices of cells partially within the geometry are adjusted to align with the boundaries of the geometry. The mesh quality might be low, however, there are tools available that allow users to verify it and identify settings that can be adjusted for improvement. After the final mesh is generated, the CFD simulation can start.

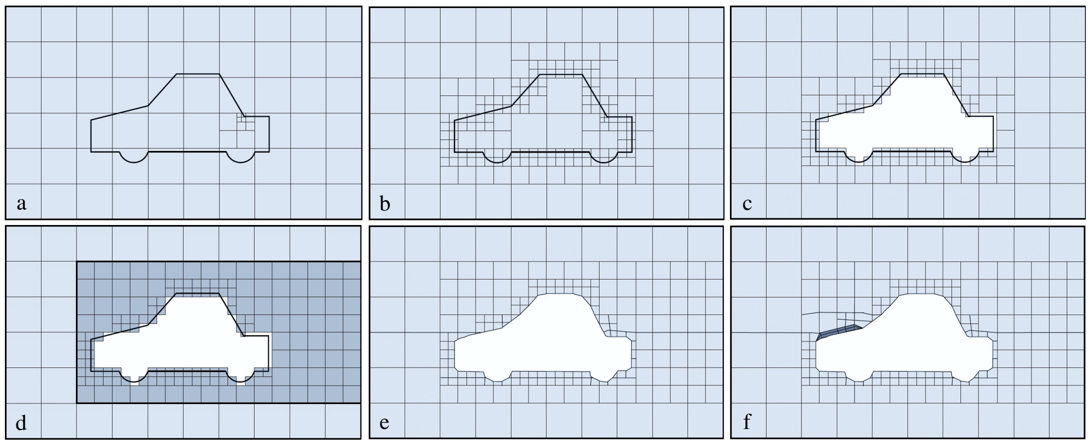


Figure 2.2.: Mesh generation in SnappyHexMesh (a) Cell division at edges, (b) Cell division at surfaces, (c) Cell removal within geometry boundaries, (d) Cell refinement in specific regions, (e) Cell vertices displacement to geometry boundaries, and (f) Mesh layers at specific locations. Adapted from "User Guide version 11" by Greenshields, C. J., 2023.

2.2. CFD guidelines for urban simulations

The following CFD guidelines for urban areas are explored. Most of them address RANS simulations, which is the most widely used method in urban physics (Blocken, 2015). As stated earlier in Section 2.1.2, this thesis also focuses on these simulations.

- Franke and Baklanov (2007) published the COST Action 732, which provides best practice guidelines (BPG) for CFD simulations in the urban environment. It is based on literature research and extracts the most general guidelines from existing ones for RANS simulations in urban and industrial environments. COST stands for the European Cooperation in Science and Technology (COST), which is a funding organisation that promotes research networks, known as COST Actions. Scientists from various disciplines have the opportunity to initiate a COST Action based on their specific interest (COST, 2023).
- A working group from the Architectural Institute of Japan (AIJ), which is a non-profit organisation for academics in architecture (AIJ), proposed CFD guidelines for pedestrian-level wind simulations. Unlike COST Action 732, these guidelines are not based on literature research, but on seven test cases for which CFD predictions, wind tunnel test results and field measurements were compared (Tominaga et al., 2008).
- Tong et al. (2016) experienced with CFD simulations on indoor-outdoor models (i.e. buildings with windows) to define suitable dimensions for the Region of Interest (RoI) for ventilation flows prediction. In contrast to the other listed guidelines, they used Large Eddy Simulations (LES).
- Existing CFD guidelines for urban areas lacked a clear definition of the Region of Interest (RoI). Liu et al. (2018) worked on a concrete approach to define this region.
- Blocken (2015) discussed some existing guidelines, and proposed some new ones to perform accurate CFD simulations for urban areas.

2.2.1. Computational domain

The computational domain is defined as "an external volumetric region that surrounds the building model, where the basic flow equations are discretized and solved" (Abu-Zidan et al., 2021). Its dimensions have a significant role in the accuracy of CFD simulation results (Franke and Baklanov, 2007; Tominaga et al., 2008; Blocken, 2015; Abu-Zidan et al., 2021), and are depended on the size of the 3D city model and the specified boundary conditions (Franke and Baklanov, 2007).

Different shapes can be used for the computational domain. While Franke and Baklanov (2007) assume a rectangular shape, their guidelines can also be applied to round and oval domains (Pađen et al., 2022). The other referenced guidelines in this section can also be followed for rectangular, oval, and round domains.

The top boundary should be placed at an adequate height to avoid local venturi effects (Abu-Zidan et al., 2021), which are flow accelerations and pressure reductions occurring in narrow areas (Oxford University Press, nd). According to Franke and Baklanov (2007), the distance between the top boundary of the computational domain and the height of the tallest building, denoted as H_{max}, should be at least $5 \cdot H_{max}$. The guidelines by Tominaga et al. (2008) state that the height must be at least $5 \cdot H_{max}$ for single buildings and depends on the terrain category of the surroundings (Tominaga et al., 2008) for urban areas. The classification is based on the location of the model regarding the city, building height, and the distance from the sea- or lakefront (Hisashi et al.).

Local venturi effects can also arise when the lateral boundaries are placed too close to the 3D urban model (Abu-Zidan et al., 2021). Franke and Baklanov (2007) suggest a distance of at least $5 \cdot H_{max}$ from the 3D city model to the boundary for single building models. For urban areas, this distance is allowed to be slightly smaller (Franke and Baklanov, 2007). However, Tominaga et al. (2008) advise to maintain $5 \cdot H_{max}$ as a minimum for single buildings, as well as for urban areas.

The approach flow enters the computational domain through the inflow boundary, situated in front of the 3D city model (Franke and Baklanov, 2007). According to Franke and Baklanov (2007), a distance of at least $5 \cdot H$ should be maintained between this boundary and the 3D city model for single building models. Blocken (2015) state that this is also true for urban models. Tominaga et al. (2008) do not specifically recommend guidelines for urban areas. However, they indicate that the location of the inlet boundary should be determined based on the upwind area covered by a smooth flow in the wind tunnel. Errors can occur in cases where this distance, also known as the upstream domain length, is insufficient. When fluid interact with buildings, their behavior change. The area covering these flows is called the wind-blocking region. An inadequate upstream domain length can lead to incorrect simulation results within this area (Abu-Zidan et al., 2021; Blocken and Carmeliet, 2006).

The wake flow can leave the computational domain through the outflow boundary, which is located behind the 3D city model. The distance between the model and the outflow boundary should be considerably greater than that between the model and the inflow boundary (Franke and Baklanov, 2007) to avoid flow recirculation (Abu-Zidan et al., 2021). Tominaga et al. (2008) recommend a distance of $10 \cdot H$ for single buildings while Franke and Baklanov (2007) advise a distance of at least $15 \cdot H$. According to Franke and Baklanov (2007), this distance can be reduced for urban areas, depending on the boundary conditions.

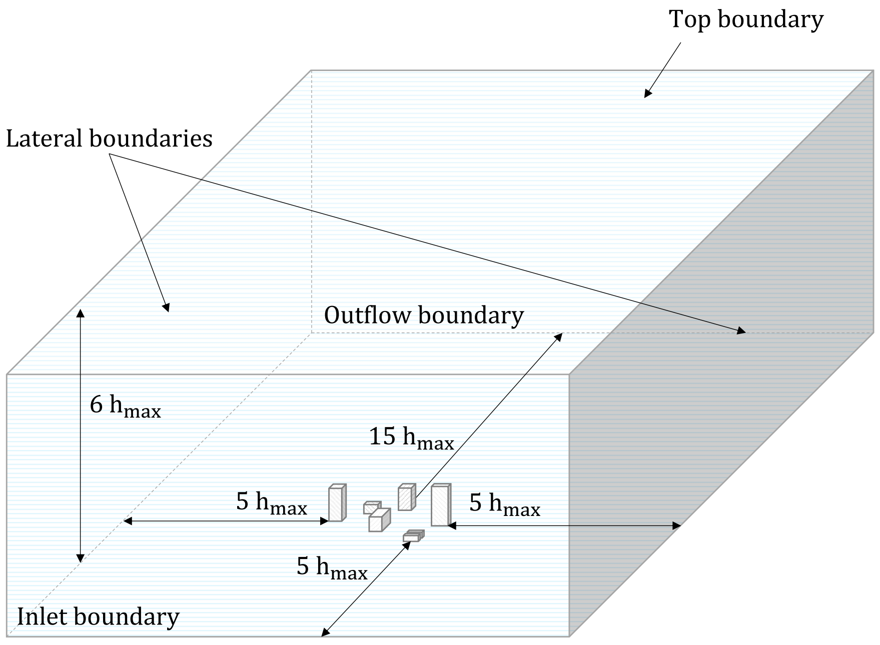


Figure 2.3.: Minimum domain dimensions adapted from Franke and Baklanov (2007).

Franke and Baklanov (2007) also suggest to use the blockage ratio (BR) defined by the Verein Deutscher Ingenieure, the Association of German Engineers, in 2005. This ratio is computed by comparing the front area of the urban model A_{urban}, the side facing the wind (inlet boundary), to the front area of the computational domain A_{domain}. The BR should remain below 3% (Franke and Baklanov, 2007; Tominaga et al., 2008) to prevent venturi effects on a global scale (Abu-Zidan et al., 2021).

$$BR = \frac{A_{urban}}{A_{domain}}, BR \leq 3\% \tag{2.11}$$

To avoid artificial accelerations in CFD simulations with large horizontal features like wide buildings or urban models, Blocken (2015) introduced a decomposition of the blockage ratio and the limit of 3% in the lateral horizontal BR_L and vertical BR_H directions. The first ratio describes the relation between the length of the urban area $L_{building}$ and the length of the domain L_{domain}. The second one defines the ratio between the height of the urban area $H_{building}$ and the height of the domain H_{domain}. These directional blockage ratios must remain below 17%, which is the square root of 3%.

$$BR_L = \frac{L_{building}}{L_{domain}}, BR_L \leq 17\% \tag{2.12}$$

$$BR_H = \frac{H_{building}}{H_{domain}}, BR_H \leq 17\% \tag{2.13}$$

Figure 2.4 illustrates the front area of the domain (inlet boundary) with the parameters needed for computing the blockage ratios.

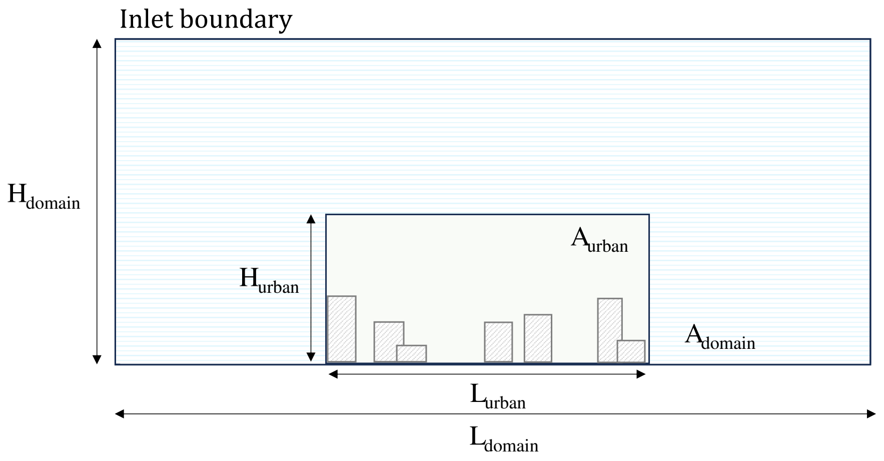


Figure 2.4.: Profile view of Figure 2.3 adapted from Franke and Baklanov (2007).

Guidelines implemented in the prototype

A selection of guidelines is considered for the prototype:

- Franke and Baklanov (2007) require a minimum distance of 5·H between the model and the boundary. Tominaga et al. (2008) suggest a less explicit definition which relies on the terrain category of the surroundings. Given that the prototype does not focus on the semantics of the 3D model, the top boundary of Franke and Baklanov (2007) is implemented.
- Both Franke and Baklanov (2007) and Tominaga et al. (2008) recommend a distance of 5·H for the lateral boundaries. While Tominaga et al. (2008) consider this distance as a minimum requirement, Franke and Baklanov (2007) allow it to be slightly shorter. Like Tominaga et al. (2008), the prototype considers this distance as a minimum to avoid local venturi effects leading to inaccurate results.
- The definition of Tominaga et al. (2008) for the inlet boundary is less concrete than the one of Franke and Baklanov (2007). Tominaga et al. (2008) rely on the upwind area covered by a smooth flow in the wind tunnel, which is not applicable in this thesis since no wind tunnels are involved. Besides, Tominaga et al. (2008) focus on single building models instead of urban models for the inlet boundary. The prototype focuses on the definition of Franke and Baklanov (2007), which uses an upstream domain length of 5·H.
- Franke and Baklanov (2007) recommend a distance of 15·H for the outlet boundary, which can be reduced for urban areas. Tominaga et al. (2008), on the other hand, suggest a distance of 10·H for single buildings. Since Franke and Baklanov (2007) define the outlet boundary for urban areas rather than single building models, the prototype focus on the guidelines of Franke and Baklanov (2007).

- The directional blockage ratios of Blocken (2015) are specifically developed to avoid simulation errors with urban models, which is not the case for the blockage ratio of Franke and Baklanov (2007). Yet, the prototype considers both the blockage ratios of Blocken (2015) and Franke and Baklanov (2007).

2.2.2. Computational grid

The domain is modelled as a continuous medium (OpenFOAM, 2015) governed by the Navier-Stokes equations, which describe the relation between "the velocity, pressure, temperature, and density of a moving fluid" (Hall, 2021). These equations are non-linear, and pose a significant challenge in solving them directly. However, discretizing the domain, also called meshing, helps to linearize these equations (OpenFOAM, 2015) and obtain a possible solution (SimScale, 2023). This discretized domain is called the computational grid, which represents "the locations in space at which model quantities are calculated, known as "grid points" or "grid cells"" (Arthur and Angevine, 2023).

OpenFOAM obtains a solution for these linearized equations by using an iterative method (Greenshields, 2023), which uses an initial value to generate a new value. The new value then replaces the initial one, and the process is repeated several times until the values converge to a final solution (Kováčová and Richtáriková, 2020). For this process, Blocken (2015) suggest to use second-order discretization schemes, instead of first-order schemes, for more accurate simulation results.

Franke and Baklanov (2007) focus on the two most common discretization methods in CFD software: the Finite Volume (FV) and Finite Difference methods (FD). The first method subdivides the domain into smaller volumes or cells, and at the center of each cell, the Navier-Stokes equations are resolved. The resulting value is attributed to the entire cell (OpenFOAM, 2010). The second method uses discrete points, for which the equations are solved. A solution is only attributed to that specific point (Shukla et al., 2011). OpenFOAM uses the Finite Volume Method (FVM) (OpenFOAM, 2010).

It is essential to generate computational grids with high quality to prevent inaccurate results and/or convergence problems. A high grid resolution and cell quality is therefore needed (Blocken, 2015). Franke and Baklanov (2007) recommends the following:

- Regular grids are preferred to irregular ones.
- The expansion ratio between two neighbouring cells must remain below 1.3 in regions of high gradients.
- The line connecting the centroids of two neighbouring cells, passing through a face f, must be parallel to the normal vector of f.

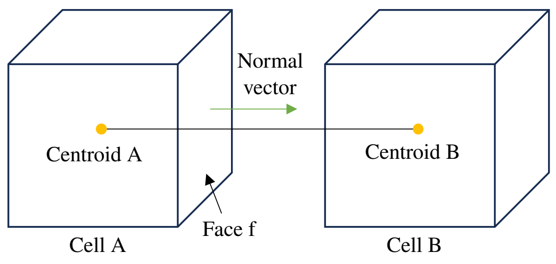


Figure 2.5.: Connection of two neighbouring cells.

- The grid lines should be perpendicularly aligned to walls (bottom boundary of the domain).
- In the Region of Interest (ROI), a minimum of 10 cells per cube root of the building volume ($\sqrt[3]{buildingvolume}$) and 10 cells between two buildings should be used as an initial minimum grid resolution. The grid must be refined to determine the correct resolution.
- Three systematically refined grids are necessary for the Richardson extrapolation, with a refinement factor of at least 3.4 (Blocken, 2015).
- The evaluation height of 1.5-2 meters should be aligned with the 3^{rd} or 4^{th} cell of the computational grid (Blocken, 2015).

Many of these guidelines are confirmed by Tominaga et al. (2008), or share similarities with their recommendations:

- They advise to use smaller cells around buildings to get more accurate flow simulations. Like the COST Action 732, they recommend a minimum of 10 cells on each side of the buildings and a stretching ratio of 1.3 or less in these areas.
- A minimum grid resolution of approximately one-tenth the scale of the building is suggested.
- The evaluation height, typically within the range of 1.5 and 5 meters above ground level, should align with the 3^{rd} cell or higher from the ground surface.
- The number of finer cells should be at least 1.5 times the number of coarser cells in each dimension, which is consistent with COST Action 732 that recommends a minimum ratio of 3.4 between two neighbouring cells ($1.5^3 = 3.375$).
- To avoid large aspect ratios of cells, they advise to use grid lines perpendicular to walls (bottom boundary of the domain) or ground surfaces.

(Tominaga et al., 2008) also stated that the first evaluation node must be placed at least the roughness height from the wall, which is the bottom boundary of the domain.

Guidelines implemented in the prototype

The guidelines of Franke and Baklanov (2007) and Tominaga et al. (2008) are quite similar, with minor differences observed in the evaluation height and linear refinement requirements. Franke and Baklanov (2007) suggest to generate the 3^{rd} or 4^{th} cell at a height of 1.5-2 meters, while Tominaga et al. (2008) recommend modeling it at a range of 1.5-5 meters. Also, the ratio between neighboring cells is 3.4 for Franke and Baklanov (2007) and 3.375 for Tominaga et al. (2008). However, OpenFOAM automatically generates a linear refinement of 2, making this guideline irrelevant for the developed prototype. Others are already considered by this software. Consequently, this thesis focuses on the following guidelines:

- Regular grid,
- A minimum of 10 cells per cube root of the building volume,
- A minimum of 10 cells between two buildings,
- A number of three refinement grids,
- An evaluation height between 1.5-5m, which is a combination of the guidelines suggested by Blocken (2015) and Tominaga et al. (2008),
- The first evaluation node placed at a minimum distance of at least the roughness height from the ground.

2.2.3. Region of interest

The level of detail of the surrounding buildings significantly impacts the simulation results around the target building (Liu et al., 2018; García-Sánchez et al., 2021). However, modelling all buildings in the domain would be computationally expensive. So, defining the surrounding buildings to be modeled to avoid poor simulation results around the target building and long execution time would be useful.

These buildings can be identified by defining a Region of Interest (RoI) or Influence Region. Tong et al. (2016) defines this region as ”the area where the surrounding buildings must be modeled explicitly to predict the ventilation flow rate accurately” in urban areas. Tominaga et al. (2008) and Liu et al. (2018) also proposed such a Region of Interest for urban wind simulations.

According to Tominaga et al. (2008), the buildings within a radius of 1 or 2 times h_{max} (height of the tallest building), denoted as H in Figure 2.6, are part of the RoI. They also recommend to add the street blocks in each direction around this region to the RoI. However, for areas outside this selected region, it is suggested to use simplified geometries or adjust the roughness instead of modeling any building.

Tominaga et al. (2008) and Tong et al. (2016) used 3D models of cubes with a specific distribution to define their guidelines on the Region of Interest (RoI). However, the structure of urban areas is much more complex. Selecting building blocks might be difficult due to the variability in building distribution within urban areas (Liu et al., 2018).

Therefore, Liu et al. (2018) worked on a more concrete approach by performing CFD simulations using different geometric models. They concluded that the buildings around the target building within a radius of at least $3 \cdot L$, where L is the maximum dimension of the target building, are within the RoI.

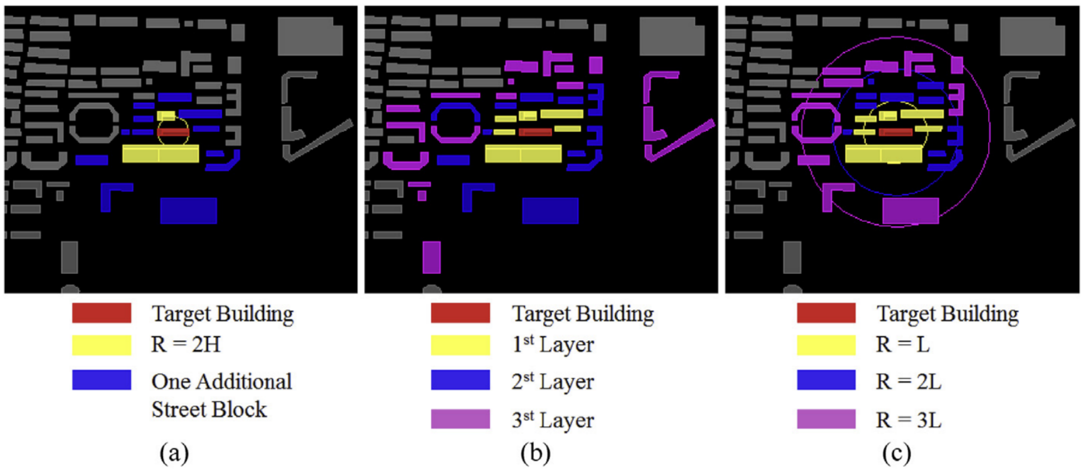


Figure 2.6.: Regions of Interest (RoI), defined by (a) Tominaga et al. (2008), (b) Tong et al. (2016), and (c) Liu et al. (2018). From "Influence of surrounding buildings on wind flow around a building predicted by CFD simulations" by Liu, et al., 2018, *Science Direct*, 96:1749–1761e.

Note that large buildings outside the domain can have a significant impact on the flow field inside the RoI. As a rule of thumb, buildings with height h_{max} within a distance of $6 \cdot h_{max}$ from the RoI should be represented in the domain (Franke and Baklanov, 2007).

Guidelines implemented in the prototype
The prototype developed in this research focuses on the guidelines defined by Liu et al. (2018) due to their more precise definition.

2.2.4. Related work

Preparing geometries for CFD simulation is a time-consuming task (Pađen et al., 2022). Therefore, Pađen et al. (2022) introduced a workflow that automatically reconstructs 3D geometries for microscale urban flow simulations from 2D geographical datasets (e.g. cadastral data, topographic datasets) and aerial pointcloud-based elevation data. This workflow considers CFD guidelines on the Region of Interest (RoI), described in Section 2.2.3, and the ones on the domain dimensions, discussed in Section 2.2.1. Buildings within the RoI are reconstructed, and the size and shape of the resulting 3D model meet the minimum requirements for the domain. The workflow led to satisfying results and was implemented in a tool called City4CFD (`https://github.com/tudelft3d/City4CFD`).

2.3. The validation of 3D geometries

2.3.1. The importance of valid geometries

3D urban models, also known as 3D city models, represent an urban environment by using 3D (vector) geometries and eventually semantics (Biljecki et al., 2015). In the past decades, these models were mainly used for visualisation; nowadays, they are increasingly used for data analysis and simulations in a wide range of domains (Biljecki et al., 2015). A challenge here is combining existing models primarily created for visualisation with GIS software, which started to specify minimum requirements to ensure correct performance. This is difficult, in particular since these requirements are not always the same. Additionally, the many definitions for 3D primitives added to this confusion (Ledoux, 2013), such as for polygons for which multiple definitions exist, even in the mathematical field (Grünbaum, 2003; Ledoux, 2013).

To increase the interoperability and exchange of geographical data, the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC) developed standards to define basic primitives (ISO19107) (ISO, 2019) and how to digitize them (OGC, 2016, 2011). Since the quality of 3D city models is crucial to perform simulations such as CFD, verifying geometries in accordance with the ISO19107 could be a ”useful starting point” (Wagner et al., 2015). However, these standards are often interpreted differently and are not always easy to implement, which leads to invalid geometries (van Oosterom et al., 2005). Therefore, tools were created identifying these errors (Section 2.3.3). Implementing them within the prototype developed in this thesis can contribute to more accurate CFD simulations, as users can improve geometries within their 3D models.

2.3.2. The ISO19107 standard and its implementation

The ISO19107 standard provides conceptual schemas and operations to describe geospatial data, including vector geometries and topology (ISO, 2019). This standard defines geometric primitives from 0D to 3D and is used to represent real-world features in GIS (Arroyo Ohori et al., 2022). Figure 2.7 shows some examples of 3D primitives along with their corresponding classification names.

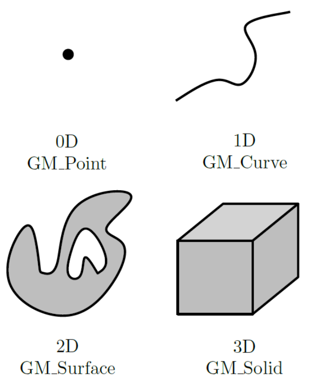


Figure 2.7.: 3D primitives based on the ISO19107 standard. From "val3dity: validation of 3D GIS primitives according to the international standards" by Ledoux (2018).

The standard ISO19107 states that a *d*-dimensional primitive is composed of (*d*-1)-dimensional primitives (Arroyo Ohori et al., 2022) and can be part of another *d*-dimensional primitive to form an aggregate or composite (Ledoux, 2018). An aggregate is a set of *d*-dimensional primitives that can overlap or be disconnected (topological relationships are not considered); a composite is a set of *d*-dimensional primitives that form a *d*-manifold, which is a shape that can be deformed to resemble/represent the *d*-dimensional Euclidean space in a continuous and invertible way (Weisstein, nd; Arroyo Ohori et al., 2022; Ledoux, 2018). In GML, these primitives are named *Multi** and *Composite** where, for example, * can be replaced by *Surface* or *Solid* (Arroyo Ohori et al., 2022; Ledoux, 2018).

In practice, curves (*GM_Curves*) and surfaces (*GM_Surfaces*) are often considered as linear or planar, respectively. With this in mind, Arroyo Ohori et al. (2022) summarised the validation rules for 3D primitives. It is worth noting that "for a three-dimensional primitive to be valid, all its lower-dimensionality primitives should also be valid" (Ledoux, 2013).

- A 2D Polygon is valid when the six assertions from OGC (2011) in Figure 2.8 are followed.

1. Polygons are topologically closed;
2. The boundary of a Polygon consists of a set of LinearRings that make up its exterior and interior boundaries;
3. No two Rings in the boundary cross and the Rings in the boundary of a Polygon may intersect at a Point but only as a tangent, eg

$$\forall P \in Polygon, \forall c1, c2 \in P.Boundary(), c1 \neq c2,$$

$$\forall p, q \in Point, p, q \in c1, p \neq q, [p \in c2 \Rightarrow q \notin c2];$$

4. A Polygon may not have cut lines, spikes or punctures eg:

$$\forall P \in Polygon, P = P.Interior.Closure;$$

5. The interior of every Polygon is a connected point set;
6. The exterior of a Polygon with 1 or more holes is not connected. Each hole defines a connected component of the exterior.

Figure 2.8.: Assertions describing a valid 2D polygon. From ”3D modelling of the built environment” volume v0.8. by Arroyo Ohori et al. (2022).

- Since a *MultiSurface* consists of *Polygons*, it is valid when each of them are valid.
- A *CompositeSurface* is valid when, similar to *MultiSurfaces*, its *Polygons* are valid. Additionally, these polygons can not overlap and/or be disjoint.
- The validity of *Solids* can be verified by generalising the six assertions developed for *Polygons* to 3D, except for the third one. In other words, *Polygon* can be replaced by *Solid*, *Ring* by *Shell*, and hole by cavity. A *Shell* represents the boundary of a *Solid* (Ledoux, 2013).
    1. ***Solids*** are topological closed;
    2. The boundary of a ***Solid*** consists of a set of ***Shell*** that make up its exterior and interior boundaries;
    3. No two ***Shells*** in the boundary cross and the Rings in the boundary of a ***Solid*** may intersect at a Point by only as a tangent;
    4. A ***Solid*** may not have cut lines, spikes or punctures;
    5. The interior of every ***Solid*** is a connected point set;
    6. The exterior of a ***Solid*** with 1 or more **cavities** is not connected. Each **cavity** defines a connected component of the exterior.
- Each *Solid* forming a *MultiSolid* must be valid.
- A *CompositeSolid* is valid when the interiors of each *Solid* are not overlapping and form a single solid after unification.

Standards help to increase the interoperability and exchange of geographical data; however, as mentioned earlier, they are often interpreted differently and are not always easy to implement. This leads to invalid geometries (van Oosterom et al., 2005). Therefore, validation tools were developed, which are discussed in the following section.

2.3.3. Related work

In contrast to 3D datasets, well-defined rules and open-source tools exist to perform geometric validations in 2D. The most known are JTS Topology Suite and GEOS (Ledoux, 2013). The former is a Java library that allows the creation and manipulation of geometries (JTS, a) and provides some validation tools (e.g. isValid() for topological checks) (JTS, b). The second started as a C++ port of JTS Topology Suite and nowadays, they still share the majority of their algorithms (GEOS).

Several software tools exist that validate 3D datasets, such as ArcGIS Pro, Oracle Spatial, and CityDoctor. Yet, they do not respect all the definitions within ISO19107 and/or do not support aggregates and composites (Ledoux, 2018; Ledoux, 2013). However, Ledoux (2018) developed open source software, named val3dity, that validates 3D primitives according to ISO19107. This tool applies the common exception, as mentioned previously, that they need to be linear or planar.

2.3.4. The open-source software val3dity

val3dity is an open-source software that validates 3D primitives based on ISO19107, with the common GIS exception that they need to be linear or planar. This tool was developed by Ledoux (2018) and is available as a command-line-only software or web application. It is compatible with the CityJSON, OBJ, OFF, POLY, and IndoorGML formats.

Figure 2.9 illustrates 3D primitives supported by val3dity. Given the requirement for primitives to be linear or planar, *LinearRings* and *Polygons* are represented as linear *GM_Curves* and planar *GM_Surfaces*, respectively. Figure 2.9 shows also that a *Solid* is formed by an external *Shell* and may contain internal *Shells*. These *Shells* can be intersected and/or disconnected within *MultiSolids*. However, *Shells* within *CompositeSolids* can only interact under the condition that the interiors of each *Solid* are non-overlapping and form a single solid after unification.

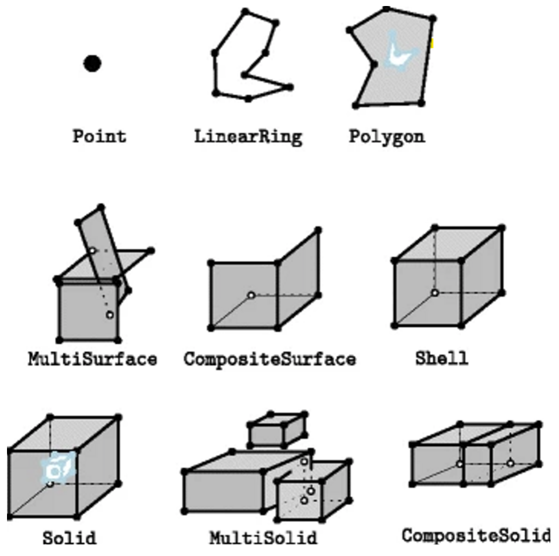


Figure 2.9.: Supported 3D primitives by val3dity. From "val3dity: validation of 3D GIS primitives according to the international standards" by Ledoux (2018).

As mentioned earlier, ISO19107 states that "for a three-dimensional primitive to be valid, all its lower-dimensionality primitives should also be valid" (Ledoux, 2013). Hence, Ledoux (2018) implemented a hierarchical validation method that, consequently, avoid "cascading" errors (errors caused by errors in primitives with lower dimensionalities). Given a *CompositeSolid*, this methods executes the following steps. First, *LinearRings* and *Polygons* are validated using planar graphs and 2D validation methods (Ledoux, 2013). Second, the validation of *Shells*, formed by these primitives, is performed. This includes testing the planarity and orientation of the surfaces of these *Shells* (Ledoux, 2013). Third, the topological relarionships between the *Shells* forming *Solids* are verified. To validate the topological relationships between these *Shells*, the concept of Nef polyhedron is used (Ledoux, 2013) that stores neighbourhood information around the vertices of these *Solids* (Arroyo Ohori et al., 2022). Finally, *CompositeSolids*, which comprises a set of *Solids*, are validated.

Figure 2.10 summarizes the errors that val3dity can identify. The majority of them correspond to primitive validations, yet, some errors are categorized under "CityGML Objects" and "Others". The first category checks whether CityGML contains geometry and non-overlapping *BuildingParts* (primitives representing buildings); the second one verifies the schema of the input file. Figure 2.10 also shows that higher dimensionalities result in more validations. For example, a *CompositeSolid* validation includes all the existing validations except those in the "CityGML Objects" and "Others" classes. However, a *MultiSurface* can be verified by the validations within the *LinearRing* and *Polygon* classes.

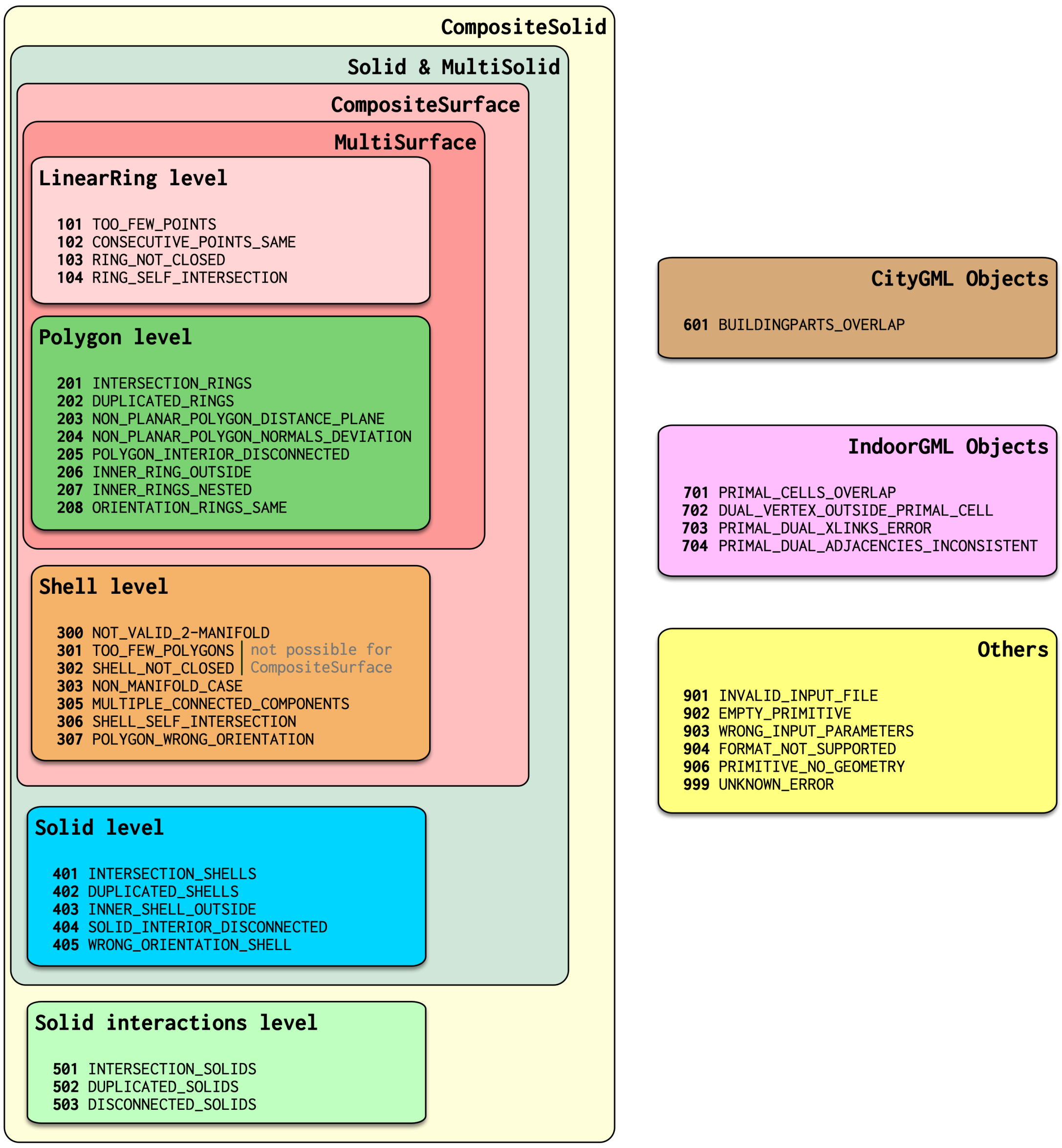


Figure 2.10.: Error codes in val3dity. From ”val3dity: validation of 3D GIS primitives according to the international standards” by (Ledoux, 2018).

Not all errors are considered critical to users, hence, val3dity takes into account some user-defined tolerances.

- Snap tolerance: Two vertices are considered identical if they are separated by this value or less (default: 0.001)
- Planarity tolerance: ISO19107 requires all vertices of a planar surface to lie on one plane, which is almost impossible with real-world data (Ledoux, 2013). For this reason, val3dity considers this parameter defining the maximum distance between a point and a fitted plane (Ledoux, 2018) (default: 0.001)

- Overlap tolerance: Two *Solids* that overlap or are disconnected by this value or less are considered properly connected (default: -1, which stands for disabled).

The topological relationship is only verified between *BuildingParts*, meaning that floating buildings over the terrain are considered as valid (Ledoux, 2018). This is an issue while running CFD simulations (e.g. simulated non-existent flows under buildings). One of the objectives of this thesis is to develop this missing validation.

2.4. 3D geometries in CFD simulations

A large number of small geometric details might affect the mesh generation (Piepereit et al., 2018), even though they are valid based on the ISO19107 standard. These geometries include sharp angles, short edges, small distances between buildings, and overlapping buildings (Pađen et al., 2022). Additionally, slivers can also lead to mesh issues (Smith, 2014). Identifying these geometries might help users simplify their geometries to improve the mesh quality and consequently, their CFD simulations.

Park et al. (2020) developed an automatic building simplification method for finite volume method (FVM), which is a discretisation method also used in OpenFOAM (Section 2.2.2). This simplification method includes merging a set of solids (forming one building) into a single solid. Additionally, faces representing details or curves are deleted and replaced. Piepereit et al. (2018) also contributed to the simplification of geometries for CFD simulations by introducing a sweep-plane algorithm that eliminates edges shorter than a given threshold.

3. Methodology

This chapter presents the methodology used to identify geometric validations and define mesh parameters. First, the chapter introduces the approach (Section 3.1.1). Subsequently, the chapter delves into developed methods for geometric validations (Section 3.2) and mesh parameters definition (Section 3.3).

3.1. Approach

3.1.1. User perspective

A prototype in the form of a web application has been developed that enables users to perform geometric validations of their 3D city models, and computes mesh parameters for CFD simulations in OpenFOAM.

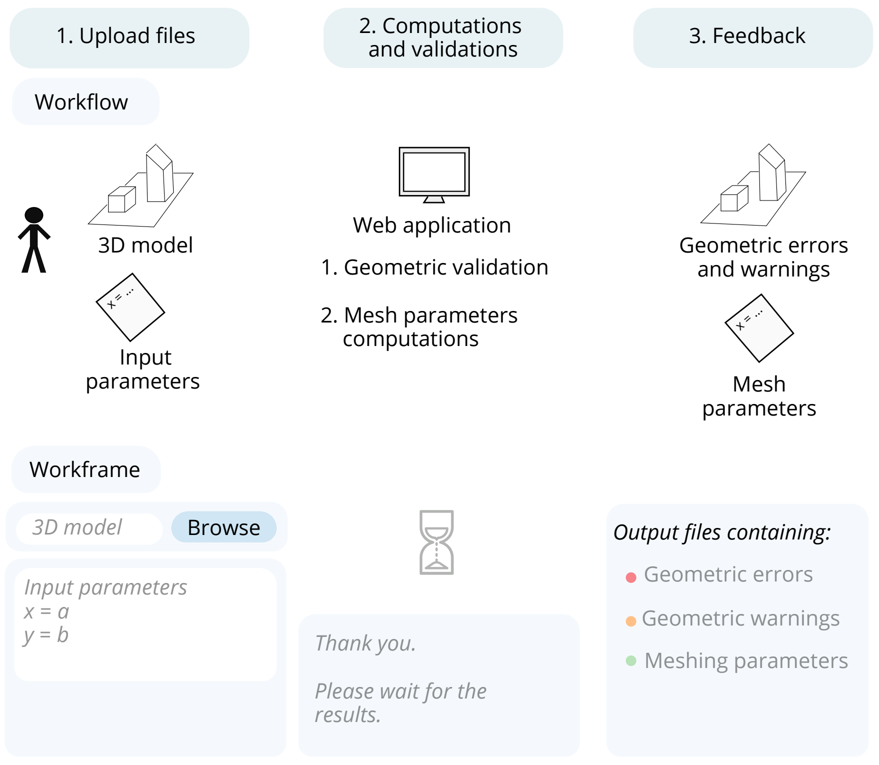


Figure 3.1.: User perspective of the prototype.

To obtain geometric validation results and CFD mesh parameters, users have to upload their 3D city model (only linear and planar geometries with triangulated faces are allowed) and specify some parameters (listed in Section 4.1). After completing the input data, the web application returns geometric errors and warnings in the form of reports and 3D models. It also generates two OpenFOAM files, *blockMeshDict* and *snappyHexMeshDict*, that contain mesh parameters according to the CFD guidelines for urban models.

3.1.2. Architecture

Figure 3.2 shows the architecture of the prototype, which comprises the following components:

- **Input data**: Users upload their 3D city model and specify parameters for geometric validations and mesh parameters definition. OBJ and STL formats with triangular faces are supported by this prototype.
- **Geometric validation**: This component can be divided into three parts:
  - Separate building and terrain validation: Validates the geometries within the 3D urban model according to the ISO19107 standard, with the common exception that they need to be linear or planar. The open source software val3dity, described in Section 2.3.4, is used to perform this validation.
  - Topological relationships between buildings and terrain validation: Checks the correct positioning of buildings on terrain to avoid incorrect simulation results. As stated in Section 2.3.4, val3dity does not verify the topological relation between buildings and terrain features. However, this is important for the CFD meshing process. Having space between buildings and terrain features would lead to incorrect simulation results. These floating buildings are reported to users. In addition, buildings that are too far below the ground line are also identified and reported to users.
  - Required validations for meshing in OpenFOAM: Identifies geometries that could lead to inappropriate meshing, such as sharp angles, short edges, sliver triangles and/or overlapping buildings.
- **CFD mesh parameters definition**: Computes mesh parameters according to the recent CFD guidelines selected in Chapter 2.2. Users receive two files, *blockMeshDict* and *snappyHexMeshDict*, that include mesh and cell dimensions. Also, information is given on how many and which buildings respect some of these CFD guidelines.
- **Output data**: Errors, warnings and mesh parameters are retrieved and reported to users.

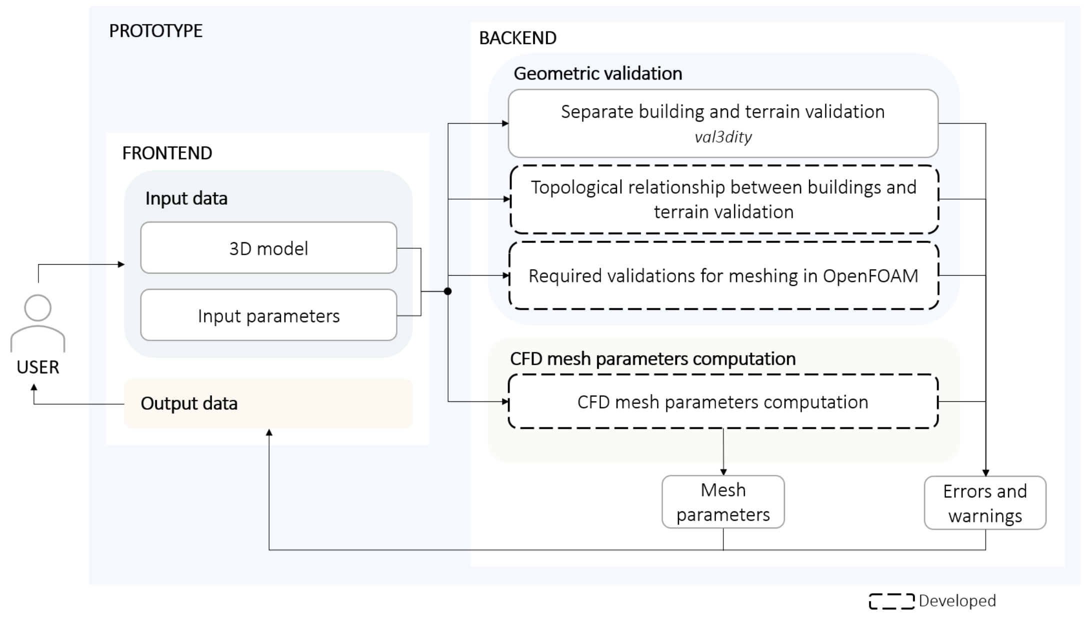

PROTOTYPE
BACKEND
Geometric validation
Separate building and terrain validation
val3dity
Topological relationship between buildings and terrain validation
Required validations for meshing in OpenFOAM
FRONTEND
Input data
3D model
Input parameters
USER
Output data
CFD mesh parameters computation
CFD mesh parameters computation
Mesh parameters
Errors and warnings
Developed

Figure 3.2.: Architecture of the prototype. The parts indicated in dashed lines were developed during this thesis.

3.2. Geometric validations

3.2.1. Separate building and terrain validation

As explained in Section 2.3, building and terrain objects must be validated based on the ISO19107 standard. Therefore, the val3dity library, which verifies whether 3D primitives are conform to this standard, is implemented within the prototype. More information on this software can be found in Section 2.3.4.

3.2.2. Topological relationships between buildings and terrain validation

The validation of topological relationships involves verifying whether building objects are correctly positioned on terrain surfaces. As described in Section 2.4, floating buildings must be avoided and, therefore, identified. In fact, these buildings result in inaccurate CFD simulation results, as in the real world, no such strong flows would pass beneath them. Also, buildings with ground surfaces placed below terrain surfaces with a distance higher than a threshold ζ are considered invalid. Figure 3.3 illustrates some possible topological relationships between building and terrain features, along with their corresponding validity status. The example shown are based on flat terrains, yet, this validation method also considers terrains with slopes.

The validation method considers two types of 3D city models: the ones with terrain surfaces, and the ones without them. The reason is that terrains are not always modeled. The method that defines ground surfaces (based on geometry) is described in Appendix B. Hence, two methods were developed:

- Method 1: with terrain
  For each building, the location of every vertex v of every ground surface f_g is analysed. A f_g is correctly positioned if all v are placed on the terrain surface f_t placed below, or located slightly below within a specific threshold ζ. By default, a predefined threshold ζ of 0.5 is applied, however, users are allowed to select another one.
- Method 2: without terrain
  Every vertex v of each ground surface must be aligned with z_t, which is the lowest z coordinate of the 3D model (z_{min}) or a user-defined ground level (z_g). v is allowed to be lower than z_t within a certain threshold ζ. Similar to the first method, this threshold has a default value of 0.5, and can be changed by users.

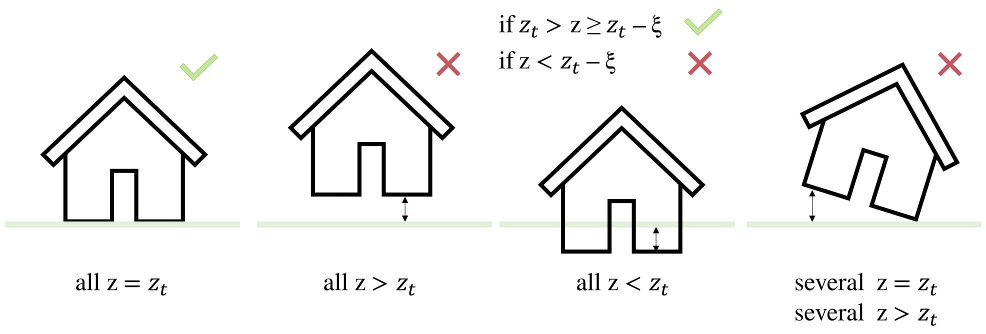


Figure 3.3.: Possible topological relationships between buildings and terrain considered by validation (z: height of the building, z_t: height of the terrain, and ζ: threshold).

For a model with terrain, Algorithm 3.1 first creates an AABB tree that stores the terrain surfaces in 2D. Then, this algorithm projects each ground surface f_g to 2D and checks if it intersects any of the surfaces f_t in the AABB tree. If there is an intersection, f_g and all corresponding f_t are converted back to 3D. The position of each vertex v of f_g relative to these f_t is then verified. If v is higher or separated from f_t by a distance greater than threshold ζ, the relationship between this v and f_t are considered invalid.
If there are no terrain surfaces, Algorithm 3.1 compares the vertices v of each ground surface f_g with z_t, which can be the lowest z-value of the model or a user-defined ground level. In other words, it compares whether buildings are on the same horizontal plane. If v is higher or separated from z_t by a distance greater than ζ, the relationship between v and z_t is considered as an error.

Algorithm 3.1: Topological relationships validation.

```
Data: buildings $B$ containing ground surfaces $f_g$ formed by vertices $v$, terrain
      surfaces $f_t$, threshold $\zeta$, and ground level $z_g$ (optional)
Result: map $topo\_errors$
1  $topo\_errors \leftarrow$ map that stores topological errors (vertices from $f_g$, surfaces $f_t$
     located below these vertices and distances between them);
2  if $terrain = true$ then
3      $m_{terrain} \leftarrow$ map (x, y) with z of vertices from every $f_t$;
4      $T_{terrain} \leftarrow$ AABB tree that stores terrain surfaces in 2D $f_{t2D}$;
5      for all $b \in B$ do
6          for all $f_g \in b$ do
7              $f_{g2D} \leftarrow f_g$ in 2D;
8              $L_{f_t} \leftarrow$ list with $f_{t2D}$ intersecting $f_{g2D}$ using $f_{g2D}$ and $T_{terrain}$;
9              for all $f_{t2D} \in L_{f_t}$ do
10                 $f_t \leftarrow$ converted $f_{t2D}$ to 3D using $m_{terrain}$;
11                 $P_{f_t} \leftarrow$ plane created with $f_t$;
12                 for all $v \in f_g$ do
13                     $position \leftarrow$ relative position between $v$ and $f_t$ (positive when $v$
                         higher than $f_t$) ;
14                     $d \leftarrow$ distance between $v$ and $f_t$;
15                     if $d > \zeta$ OR $position = positive$ then
16                         add $v$, $f_t$ and $d$ to $topo\_errors$;
17 else
18     $z_t \leftarrow$ z value at which the ground surfaces of the buildings should be placed
         ($z_{min}$);
19     if $z_g = false$ then
20         $z_t \leftarrow$ lowest z value of the buildings;
21     else
22         $z_t \leftarrow z_g$;
23     for all $b \in B$ do
24         for all $f_g \in b$ do
25             for all $v \in f_g$ do
26                 $z \leftarrow$ z value of $v$;
27                 if $z < (z_t - \zeta)$ OR $z > z_t$ then
28                     $x \leftarrow$ x value of $v$;
29                     $y \leftarrow$ y value of $v$;
30                     $v_z \leftarrow$ vertex $(x, y, z_t)$ ;
31                     $d \leftarrow$ distance between $v$ and $v_z$;
32                     add $v$, $f_t$ and $d$ to $topo\_errors$;
```

3.2.3. Required validations for meshing in OpenFOAM

Sharp angles, sliver triangles, and short edges might affect the quality of the mesh generated in OpenFOAM, which lead to inaccurate CFD simulation results (Section 2.4). For this reason, validation methods were developed to identify these geometries.

Sharp angles

In this context, sharp angles are angles between two faces smaller than a specified threshold θ. To identify these sharp angles, the prototype calculates the dihedral angle ϕ between each face f and a neighbouring face n. Two faces are neighbours if they have two vertices in common and do not share the other two vertices. To achieve this, the prototype first determines the common vertices shared by two faces (P and p_0 as illustrated in Figure 3.4), as well as the vertices unique to each face (p_1 and p_2 in Figure 3.4). Next, three vectors, namely $\vec{v}0$, $\vec{v}1$ and $\vec{v}2$ are defined to facilitate the computation of ϕ, which can be calculated using the following formula:

$$\phi = \arccos \frac{(\vec{b}0 \times \vec{b}1) \cdot (\vec{b}0 \times \vec{b}2)}{|\vec{b}0 \times \vec{b}1||\vec{b}1 \times \vec{b}2|} \tag{3.1}$$

If ϕ or $(360 - \phi)$ are found to be less than or equal to a specified threshold angle θ, expressed in degrees, the two faces are defined as sharp angles. The default value of θ is set to 2°, but can be modified by users.

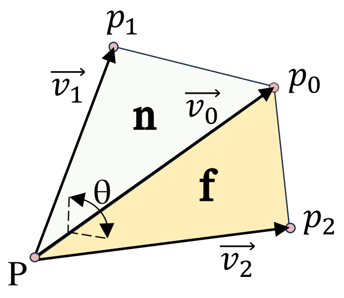


Figure 3.4.: Dihedral angle.

Algorithm 3.2: Sharp angles identification.

```
Data: building faces $f_b$ with their neighbouring faces $n$ and threshold angle $\theta$
Result: vector $sharp\_angles$
1  $sharp\_angles \leftarrow$ vector that stores sharp angles (two faces, and angle between them);
2  for all $f \in f_b$ do
3      $A_f \leftarrow$ area of f;
4      if $A_f > 0$ then
5          for all $n \in f$ do
6              $A_n \leftarrow$ area of f;
7              if $A_n > 0$ then
8                  $\phi_1 \leftarrow$ dihedral angle between $f$ and $n$;
9                  $\phi_2 \leftarrow$ 360-$\phi_1$;
10                 if $\phi_1 \leq \theta$ then
11                     $pair \leftarrow$ pair containing $f$, $n$ and $\phi_1$;
12                     if $pair \notin sharp\_angles$ then
13                         add $pair$ to $sharp\_angles$;
14                 if $\phi_2 \leq \theta$ then
15                     $pair \leftarrow$ pair containing $f$, $n$ and $\phi_2$;
16                     if $pair \notin sharp\_angles$ then
17                         add $pair$ to $sharp\_angles$;
```

Sliver triangles

Sliver triangles are very thin triangles, as illustrated in Figure 3.5. To detect them, a sliver parameter SP was defined by using the following formula (Artwork Conversion Software, nd):

$$SP = \frac{2 \times area}{perimeter} \tag{3.2}$$

SP is computed for each face within the 3D city model (including triangular faces). When SP is lower than or equal to a certain threshold σ, the corresponding face is considered a sliver. The default value of this threshold is set to 0.005, but can also be user-defined. Note that faces with collinear vertices are not identified as sliver triangles. These geometries are already considered as errors by the val3dity tool (Section 3.2.1).

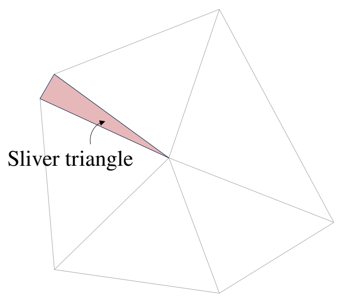


Figure 3.5.: Example of a sliver triangle.

Algorithm 3.3: Sliver triangles identification.

```
   Data: building faces $f_b$ and sliver threshold $\sigma$
   Result: vector $sliver\_triangles$
1  $sliver\_triangles \leftarrow$ vector storing faces with their sliver parameter $s$;
2  for all $f \in f_b$ do
3      $SP \leftarrow$ sliver parameter of $f$;
4      $v_1 \leftarrow$ vertex 1 of $f$;
5      $v_2 \leftarrow$ vertex 2 of $f$;
6      $v_3 \leftarrow$ vertex 3 of $f$;
7      if $SP \leq \sigma$ and $f \notin sliver\_triangles$ and $v_1, v_2, v_3 \neq collinear$ then
8          add $f$ and $SP$ to $sliver\_triangles$;
```

Short edges

Short edges are edges shorter than a specific threshold λ. To detect these short edges, the length l of every edge e that forms a face f within a building b is computed. If the length l is found to be shorter than or equal to the defined threshold λ, e is defined as a short edge. Edges with a l value of 0 (i.e. edges with identical vertices) are not identified as short edges, since the val3dity tool (Section 3.2.1) already considers them as errors.

Algorithm 3.4: Short edges identification.

```
   Data: building faces $f_b$ formed by edges $e$ and threshold length $\lambda$
   Result: vector $short\_edges$
1  $short\_edges \leftarrow$ vector storing edges with their length $l$;
2  for all $f \in f_b$ do
3      for all $e \in f$ do
4          $l \leftarrow$ length of $e$;
5          if $l < \lambda$ and $e \notin short\_edges$ and $l > 0$ then
6              add $e$ and $l$ to $short\_edges$;
```

Overlapping buildings

As discussed in Section 2.4, overlapping buildings might be an issue during the CFD meshing process. Therefore, Algorithm 3.5 identifies overlapping buildings. First, this algorithm checks for each building whether it intersects another building. If an intersection is found, the algorithm computes its volume. If the calculated volume is zero, the two buildings are considered non-overlapping; otherwise, they are regarded as overlapping. For this method, two conditions must be satisfied: the buildings must bound a volume, and must not be self-intersecting. In cases where these conditions are not met, a second method is applied, which uses the ground surfaces of these buildings. This alternative method checks if there are shared areas in 2D between the ground surfaces of these two buildings. If such a shared region exists and has an area greater than zero, the buildings are identified as overlapping. The intersections used in each method are illustrated in Figure 3.6.

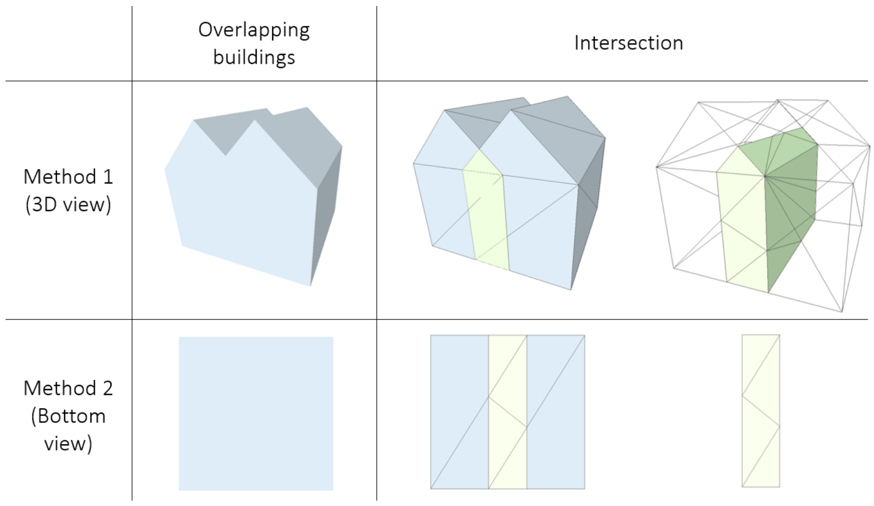


Figure 3.6.: Intersections used in methods implemented in Algorithm 3.5.

Algorithm 3.5: Overlapping buildings identification.

```
Data: buildings $B$
Result: vector $overlapping\_buildings$
$overlapping\_buildings \leftarrow$ vector storing pairs of building that overlap;
for all $b \in B$ do
    for all $bb \in B$ do
        if $b \neq bb$ then
            $vol_b \leftarrow$ volume of $b$;
            $vol_{bb} \leftarrow$ volume of $bb$;
            if $vol_b$ AND $vol_{bb} > 0$ then
                if $b$ AND $bb \neq$ self-intersecting then
                    $i \leftarrow$ intersection between $b$ and $bb$;
                    $vol_i \leftarrow$ volume of $i$;
                    if $vol_i > 0$ then
                        add $b$ and $bb$ to $overlapping\_buildings$;
        else
            $A \leftarrow$ total area of intersection ground surfaces of $b$ and $bb$;
            $gs_b \leftarrow$ ground surfaces of $b$;
            $gs_{bb} \leftarrow$ ground surfaces of $bb$;
            for all $f \in gs_b$ do
                for all $ff \in gs_{bb}$ do
                    $i \leftarrow$ intersection between $f$ and $ff$;
                    $a \leftarrow$ area of $i$;
                    add $a$ to $A$;
            if $A > 0$ then
                add $b$ and $bb$ to $overlapping\_buildings$;
```

3.3. Preparation for CFD simulation steps

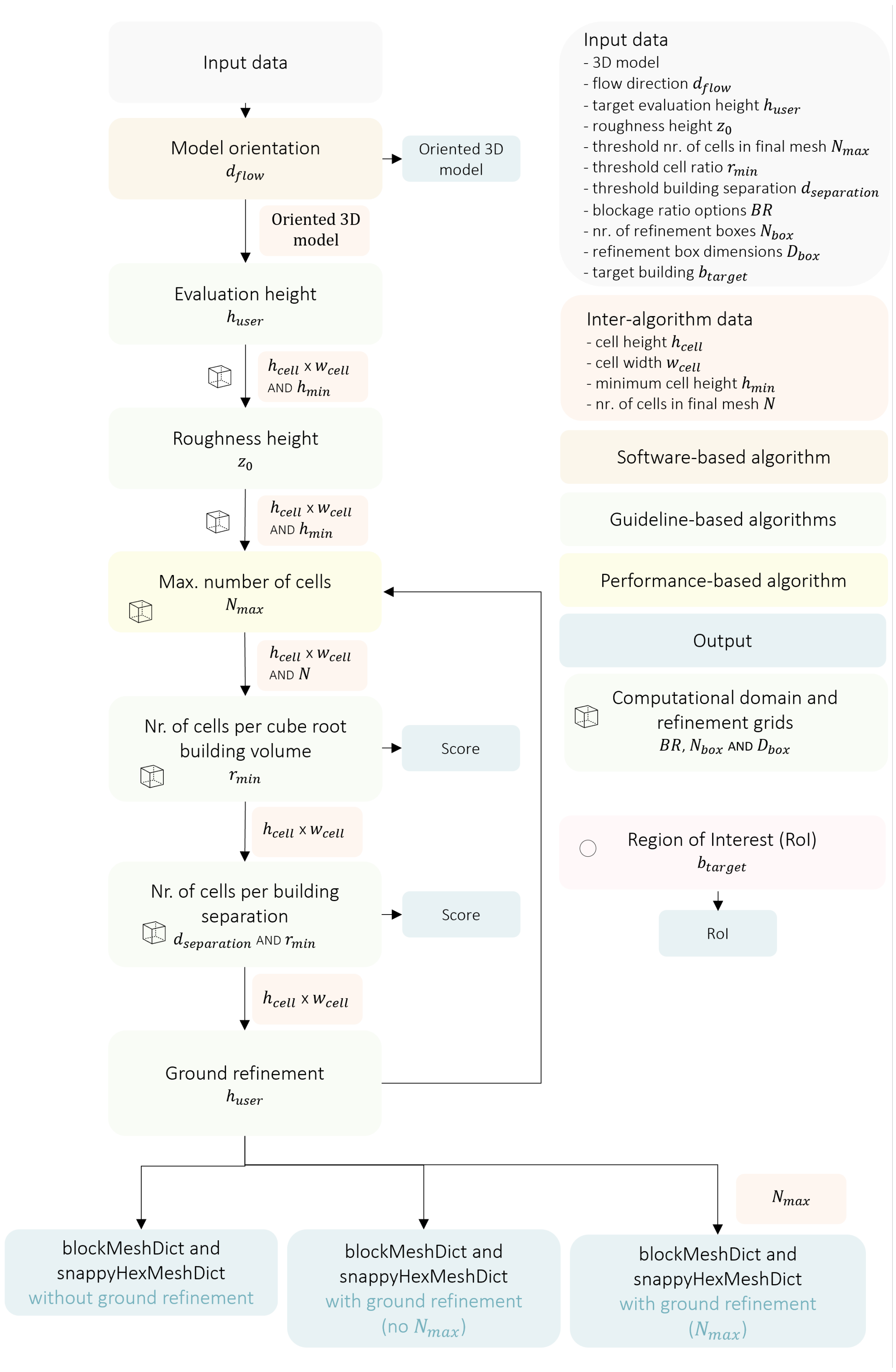
3.3.1. Overview

Figure 3.7 shows the workflow of the mesh parameters definition method for CFD simulations in urban areas. The blocks of the workflow can be classified into input data, algorithms, inter-algorithm data, and output data.

First, the method needs the input data listed on the upper right corner of Figure 3.7. Subsequently, a set of algorithms are executed. The majority of them are based on CFD guidelines except for two: the first algorithm aligns the input model with the incoming flow in OpenFOAM, and the fourth algorithm prevents the final mesh from getting too computationally expensive. For each of these algorithms, the data generated or modified by one algorithm and reused by another one is showed. The most exchanged inter-algorithm data is the cell dimensions. It is worthwhile to note that at each time the cell dimensions are modified, the domain and refinement boxes dimensions are adjusted. This is to ensure that an integer number of cells can fit in each of them. Finally, Figure 3.7 shows that data intended for users are generated. These output data comprise:

- A 3D city model (OBJ) oriented for CFD simulation in OpenFOAM,
- Buildings that satisfy the required minimum number of cells,
- The Region of Intereset (RoI),
- *blockMeshDict* files from OpenFOAM, which contains domain/background mesh information,
- *snappyHexMeshDict* files from OpenFOAM, which includes specifications for the refinement boxes.

The mesh parameters definition method is further explained in the following sections.



Input data
Model orientation d_{flow}
Oriented 3D model
Oriented 3D model
Evaluation height h_{user}
h_{cell} x w_{cell} AND h_{min}
Roughness height z_0
h_{cell} x w_{cell} AND h_{min}
Max. number of cells N_{max}
h_{cell} x w_{cell} AND N
Nr. of cells per cube root building volume r_{min}
Score
h_{cell} x w_{cell}
Nr. of cells per building separation $d_{separation}$ AND r_{min}
Score
h_{cell} x w_{cell}
Ground refinement h_{user}
blockMeshDict and snappyHexMeshDict without ground refinement
blockMeshDict and snappyHexMeshDict with ground refinement (no N_{max})
blockMeshDict and snappyHexMeshDict with ground refinement (N_{max})
N_{max}
Input data
- 3D model
- flow direction d_{flow}
- target evaluation height h_{user}
- roughness height z_0
- threshold nr. of cells in final mesh N_{max}
- threshold cell ratio r_{min}
- threshold building separation $d_{separation}$
- blockage ratio options BR
- nr. of refinement boxes N_{box}
- refinement box dimensions D_{box}
- target building b_{target}
Inter-algorithm data
- cell height h_{cell}
- cell width w_{cell}
- minimum cell height h_{min}
- nr. of cells in final mesh N
Software-based algorithm
Guideline-based algorithms
Performance-based algorithm
Output
Computational domain and refinement grids BR, N_{box} AND D_{box}
Region of Interest (RoI) b_{target}
RoI

Figure 3.7.: Workflow of the mesh parameters definition method.

3.3.2. Model orientation

The first step is to align the 3D city model with the incoming flow simulated in OpenFOAM. Therefore, Algorithm 3.6 aligns the user-defined flow with the x-axis in OpenFOAM by using the rotation matrix along the z-axis (Equation 3.3). Figure 3.8 illustrates the angles considered by this algorithm (a), and shows some examples of the orientation process (b). Similar to the angles in this figure, users must specify their flow directions in degrees. The resulting model is used throughout the mesh parameters definition method, and must be used to perform the CFD simulation. For this reason, the oriented model is made available to users.

$$R_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.3}$$

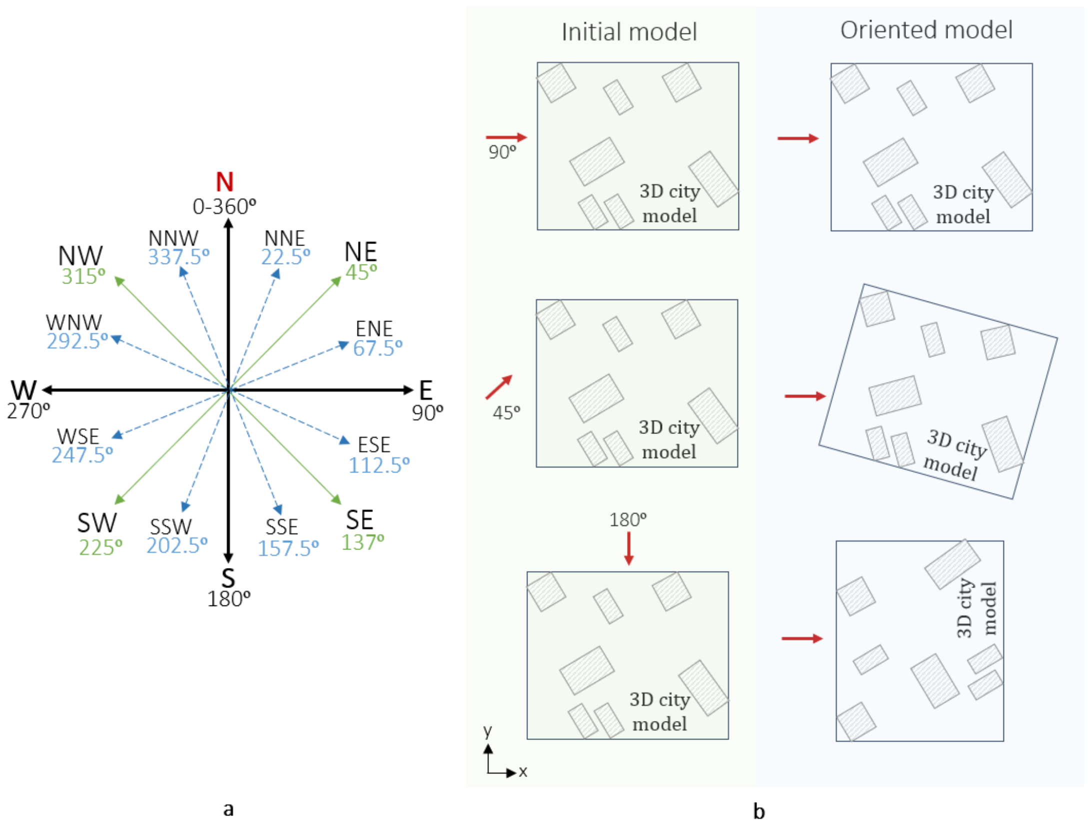


Figure 3.8.: (a) Flow directions considered by Algorithm 3.6 (b) Example of 3D city model orientations.

Algorithm 3.6: Model orientation.

```
Data: 3D city model $citymodel$ with vertices $V$ and user-defined flow direction $d_{flow}$
Result: 3D city model with the correct orientation $rotated\_citymodel$
1  $V_{rotated} \leftarrow$ vector with rotated vertices;
2  for $all\ v \in V$ do
3      $x \leftarrow$ x-coordinate of $v$;
4      $y \leftarrow$ y-coordinate of $v$;
5      $z \leftarrow$ z-coordinate of $v$;
6      $f \leftarrow ((d_{flow} - 90) \cdot \pi)/180$;
7      $x_{rotated} \leftarrow x \cdot \cos f - y * \sin f$;
8      $y_{rotated} \leftarrow x \cdot \sin f - y * \cos f$;
9      $p_{rotated} \leftarrow point(x_{rotated}, y_{rotated}, z)$;
10     add $p_{rotated}$ to $V_{rotated}$;
11 $rotated\_citymodel \leftarrow$ new OBJ file with the rotated vertices;
```

3.3.3. Evaluation height

The initial cell dimensions are based on guidelines for evaluation heights, which are heights at which flows are analysed. For CFD simulations at pedestrian level, these evaluation heights are usually set between 1.5 and 2m. For drones, for example, these could be set at 10m. The prototype considers a combination of two sets of guidelines: Franke and Baklanov (2007) suggest an evaluation height of 1.5-2m at the 3^{rd} or 4^{th} cell from the ground, and Tominaga et al. (2008) recommend this height to be 1.5-5m at the 3^{rd} cell or higher. It uses an evaluation height ranging from a minimum h_{min} (default: 1.5m), which is the user-defined target height h_{user}, to a maximum of 5m h_{max} (Figure 3.9). However, both h_{min} and h_{max} are set to the user-defined target height, when this value is higher than 5.

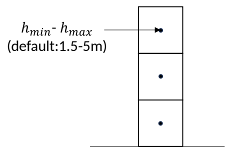


Figure 3.9.: Evaluation height used in the prototype.

Algorithm 3.7 was developed to define initial cell dimensions. It assigns h_{max} as the initial cell height h_{cell}. Since maintaining a cell ratio of approximately 1 is beneficial (as explained in Section 2.2.1), the initial cell width w_{cell} was set to the same value as h_{cell}. In addition, the algorithm stored the minimum height h_{min} to avoid the h_{cell} from being lower than required during the following steps. Finally, the algorithm defines the initial computational domain and refinement boxes, which are described in Sections 3.3.4 and 3.3.5, respectively. Figure 3.10 summarizes the defined parameters after running this algorithm.

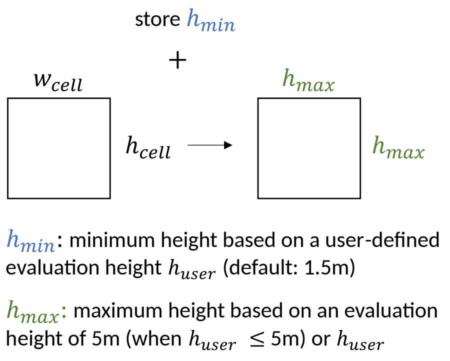

store h_{min}
+
w_{cell}
h_{cell}
h_{max}
h_{max}
h_{min}: minimum height based on a user-defined evaluation height h_{user} (default: 1.5m)
h_{max}: maximum height based on an evaluation height of 5m (when h_{user} $\leq$ 5m) or h_{user}

Figure 3.10.: Initial cell dimensions based on the evaluation height.

Algorithm 3.7: Evaluation height (2 refinement boxes).

```
    Data: user-defined evaluation height $h_{user}$
    Result: initial cell dimensions $w_{cell}$x$h_{cell}$, and minimum cell height $h_{min}$
1   $h_{min} \leftarrow (h_{user}/2.5) \cdot 8$;
2   if $h_{user} \leq 5$ then
3       $h_{max} \leftarrow (5/2.5) \cdot 8$;
4   else
5       $h_{max} \leftarrow (h_{user}/2.5) \cdot 8$;
6   $h_{cell} \leftarrow h_{max}$;
7   $w_{cell} \leftarrow h_{cell}$;
8   define the initial computational domain;
9   define initial refinement boxes;
```

3.3.4. Computational domain

The dimensions of the computational domain were defined based on the guidelines of Franke and Baklanov (2007), which suggest distances between the 3D city model and domain boundaries as illustrated in Figure 3.11. More information about these guidelines are given in Section 2.2.1. As described in Section 2.1.3, OpenFOAM subdivides this domain into hexahedral cells. The resulting domain is referred to as background mesh, and must contain a whole number of cells.

Algorithm 3.8 was written to determine the domain dimensions described by Franke and Baklanov (2007) and define the background mesh. Figure 3.11 shows that the domain dimensions depend on the height of the tallest building h_{max}. Consequently, the initial step involved identifying this height within the 3D city model. Next, this height was used to compute the x, y, z coordinates of the domain. Finally, the dimensions were adjusted to ensure an integer number of cells fit within the domain.

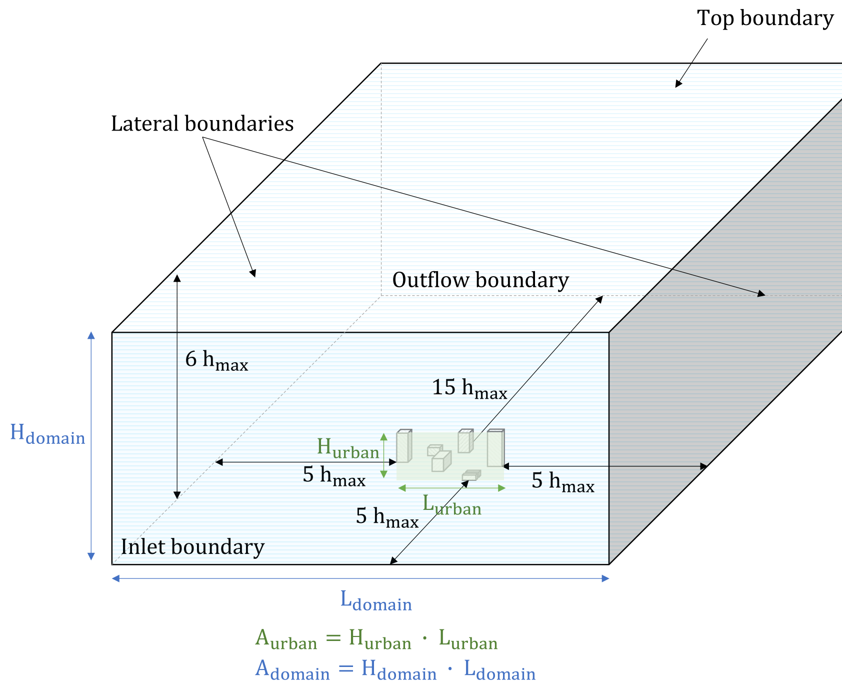


Figure 3.11.: Minimum computational domain dimensions adapted from Blocken (2015).

The prototype allows users to consider two additional set of guidelines within the computational domain definition. The first set describes a blockage ratio BR between the flow-facing area of the 3D city model, and the area of the inlet boundary of the domain (Franke and Baklanov, 2007). BR should remain below 3%. The second set suggests two blockage ratios: the lateral horizontal blockage ratio BR_L, and the vertical blockage ratio BR_H. BR_L is the ratio between the length of the urban area facing the flow direction and the length of the inlet boundary; and BR_H is the one between the height of the tallest building and the height of the computational domain. Both BR_L and BR_H must remain below 17%.

Algorithms 3.9 and 3.10 were developed for both of these additional guidelines, and are executed after Algorithm 3.8. Users are allowed to choose one of these optional guidelines. Algorithm 3.9 involves verifying whether BR is respected, and incrementally increases the length and height of the inlet boundary until BR is below 3% (Figure 3.12). Algorithm 3.10 ensures that the ratios BR_L and BR_H do not exceed 17% by iteratively increasing the length and/or the height of the inlet boundary until this limit is reached (Figure 3.13). Finally, the algorithms check whether an integer number of cells can be inserted into the domain, and slightly adjust the domain coordinates if necessary.

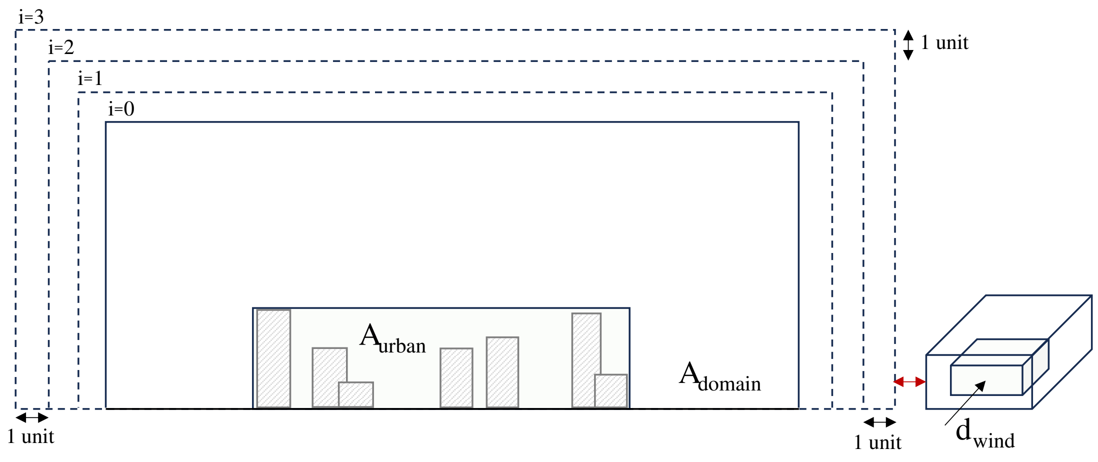


Figure 3.12.: Adjustment of domain dimensions after each iteration i based on blockage ratio BR performed by Algorithm 3.9.

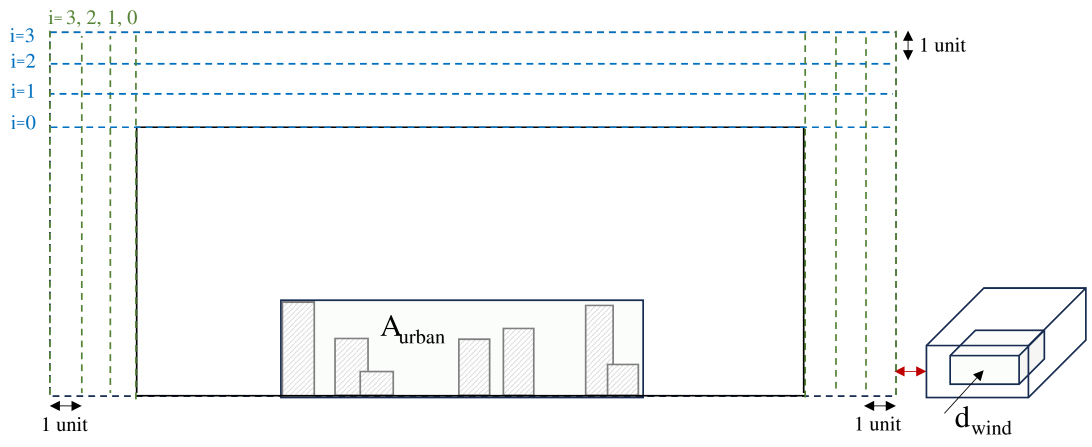


Figure 3.13.: Adjustment of domain dimensions after each iteration i based on directional blockage ratios BR_H (blue) and BR_L (green) performed by Algorithm 3.10.

Algorithm 3.8: Computational domain without blockage ratios.

```
Data: 3D city model citymodel with extremities $(cx_{min}, cy_{min}, cz_{min})$ and
      $(cx_{max}, cy_{max}, cz_{max})$, and cell dimensions $w_{cell}$x$h_{cell}$
Result: boundary box domain
1 $h_{max} \leftarrow$ height of the tallest building within the citymodel;
2 $dx_{min} \leftarrow cx_{min} - 5 * h_{max}$;
3 $dx_{max} \leftarrow cx_{max} + 15 * h_{max}$;
4 $dy_{min} \leftarrow cy_{min} - 5 * h_{max}$;
5 $dy_{max} \leftarrow cy_{max} + 5 * h_{max}$;
6 $dz_{min} \leftarrow cz_{min}$;
7 $dz_{max} \leftarrow cz_{max} + 5 * h_{max}$;
8 adjust $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to fit a whole number of cells
    with dimensions $w_{cell}$x$h_{cell}$ within this region;
9 create a bounding box domain with extreme points $(dx_{min}, dy_{min}, dz_{min})$ and
    $(dx_{max}, dy_{max}, dz_{max})$;
```

Algorithm 3.9: Computational domain with *BR*.

```
Data: 3D city model citymodel, boundary box domain with extremities
      $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$, and cell dimensions $w_{cell}$x$h_{cell}$
Result: boundary box domain
1  $A_{urban} \leftarrow$ front area of citymodel;
2  $A_{domain} \leftarrow$ front area of domain;
3  $BR \leftarrow A_{urban} / A_{domain}$;
4  while $BR > 3\%$ do
5      $dy_{min} \leftarrow dy_{min} - 0.01$;
6      $dy_{max} \leftarrow dy_{max} + 0.01$;
7      $dz_{min} \leftarrow dz_{min} - 0.01$;
8      $dz_{max} \leftarrow dz_{max} + 0.01$;
9      update $BR$;
10 adjust $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to fit a whole number of cells
     with dimensions $w_{cell}$x$h_{cell}$ within this region;
11 assign new extremity values $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to
     domain;
```

Algorithm 3.10: Computational domain with BR_H and BR_L.

```
Data: 3D city model $citymodel$, boundary box $domain$ with extremities
      $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$, and cell dimensions $w_{cell}$x$h_{cell}$
Result: boundary box $domain$
1  $L_{urban} \leftarrow$ length of $citymodel$;
2  $L_{domain} \leftarrow$ length of $domain$;
3  $BR_L \leftarrow L_{urban} / L_{domain}$;
4  $H_{urban} \leftarrow$ height of $citymodel$;
5  $H_{domain} \leftarrow$ height of $domain$;
6  $BR_H \leftarrow H_{urban} / H_{domain}$;
7  while $BR_L > 17\%$ do
8      $dy_{min} \leftarrow dy_{min} - 0.01$;
9      $dy_{max} \leftarrow dy_{max} + 0.01$;
10     update $BR_L$;
11 while $BR_H > 17\%$ do
12     $dz_{min} \leftarrow dz_{min} - 0.01$;
13     $dz_{max} \leftarrow dz_{max} + 0.01$;
14     update $BR_H$;
15 adjust $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to fit a whole number of cells
     with dimensions $w_{cell}$x$h_{cell}$ within this region;
16 assign new extremity values $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to
     $domain$;
```

3.3.5. Refinement boxes

As mentioned earlier in Section 2.2.2, a minimum number of three refinement boxes is suggested for CFD simulations in urban areas. However, two refinement boxes are often used in practise. That is why the prototype allows users to chose between two or three refinement boxes. A refinement box delineates an area where the cell dimensions are reduced to increase the number of cells in which the Navier-Stoke equations can be solved. This cell refinement increases the resolution within this area, and consequently enhances the accuracy of the simulation.

In general, with three refinement boxes, the first box is placed directly around the urban area and is a bit higher (e.g. half of h_{max}) than the tallest building h_{max}; the second box surrounds the first and extends slightly toward the outflow boundary; the last box is placed around the second one. With two refinement boxes, only the two first of these boxes are generated. Figure 3.14 shows an example of refinement boxes defined for an urban CFD simulation. The refinement boxes should have the same proportions as the computational domain. Like the domain, the refinement box dimensions are thus determined by the distance between their boundaries and the urban model. These distances are multiples of the height of the tallest building h_{max}. Table 3.1 shows these multiples used as default values. However, users are allowed to change them.

Table 3.1.: h_{max} multiples used to define default refinement box dimensions.

Boundary:	Inlet	Outlet	Lateral	Top
Box 1	0.5	0.5	0.5	1.5
Box 2	2	6	2	2
Box 3	3	9	3	3

A refinement level must be specified for each refinement box. In OpenFOAM, integers can be used to specify refinement levels. Level 0 corresponds to cells without refinement, and with each subsequent refinement level, a cell is divided into 8 smaller cells. Table 3.2 provides an example of cell dimensions per refinement level, assuming a cell ratio of 1.
As illustrated in Figure 3.14, the closer the refinement box is to the urban area, the higher the refinement level of the cells. Given that the domain contains cells without refinement, and the prototype considers cells with a refinement level of 4 around the buildings when three refinement boxes are used, box 1, box 2, and box 3 have refinement levels 3, 2, and 1, respectively. With two refinement boxes, the level around the buildings is 3. Also, the levels in boxes 1 and 2 are 1 and 2, respectively.

Table 3.2.: Example of cell dimensions per refinement level (cell ratio is 1).

Refinement level	Cell width	Cell volume	Example
0	w_{cell}	w_{cell}^3	$w_{cell} = 8$
1	$w_{cell}/2$	$(w_{cell}/2)^3$	$w_{cell} = 4$
2	$w_{cell}/4$	$(w_{cell}/4)^3$	$w_{cell} = 2$
3	$w_{cell}/8$	$(w_{cell}/8)^3$	$w_{cell} = 1$
4	$w_{cell}/16$	$(w_{cell}/16)^3$	$w_{cell} = 0.5$

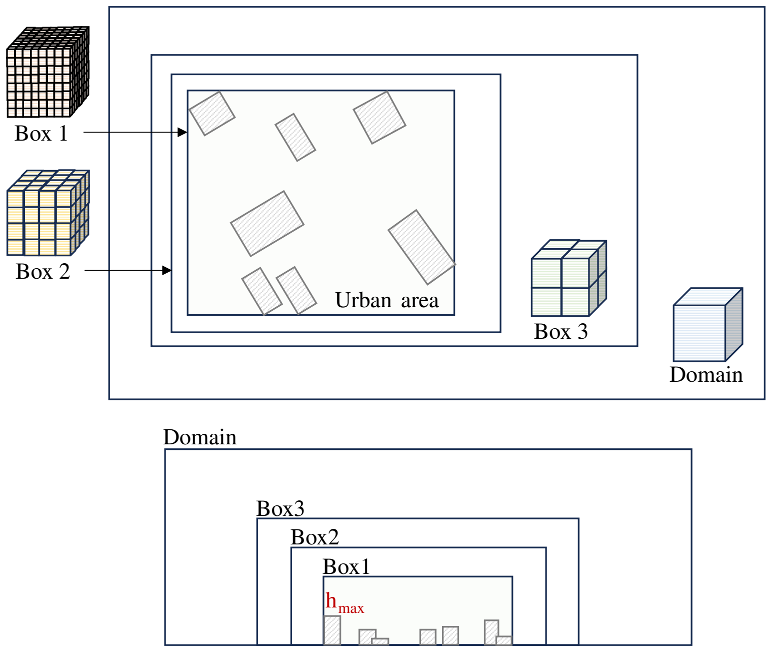


Figure 3.14.: Example of refinement boxes.

Algorithm 3.11 was executed for each refinement box. To compute the coordinates of the boxes, it uses the multiple of the height of the tallest building, as explained above. Then, the dimensions were adjusted to ensure an integer number of cells (with the correct refinement level) fits within the box.

Algorithm 3.11: Refinement box.

```
Data: 3D city model $citymodel$ with extremities $(cx_{min}, cy_{min}, cz_{min})$ and
      $(cx_{max}, cy_{max}, cz_{max})$, cell dimensions $w_{cell}$x$h_{cell}$, height tallest building $h_{max}$,
      and factors $finlet$, $foutlet$, $flateral$, $ftop$ ($h_{max}$ multiples in each direction)
Result: boundary box $refinementbox$
1 $x_{min} \leftarrow cx_{min} - finlet * h_{max}$;
2 $x_{max} \leftarrow cx_{max} + foutlet * h_{max}$;
3 $y_{min} \leftarrow cy_{min} - flateral * h_{max}$;
4 $y_{max} \leftarrow cy_{max} + flateral * h_{max}$;
5 $z_{min} \leftarrow cz_{min}$;
6 $z_{max} \leftarrow cz_{max} + ftop * h_{max}$;
7 adjust $(dx_{min}, dy_{min}, dz_{min})$ and $(dx_{max}, dy_{max}, dz_{max})$ to fit a whole number of cells
    with dimensions $w_{cell}$x$h_{cell}$ within this region;
8 create a bounding box $domain$ with extreme points $(dx_{min}, dy_{min}, dz_{min})$ and
    $(dx_{max}, dy_{max}, dz_{max})$;
```

3.3.6. Roughness height

The centre point of the first cells from the bottom boundary should be placed at a minimum distance of at least one roughness height z_0 from the wall (bottom boundary)(Tominaga et al., 2008), which is the bottom boundary of the computational domain. Consequently, the cell dimensions h_{cell}xw_{cell} and the minimum cell height h_{min} may have to be redefined. To verify whether this is the case, Algorithm 3.12 computes the minimum cell height h_r to satisfy this condition and then compares h_r to the current h_{min} and h_{cell}. Figure 3.15 shows several scenarios that might occur. In short, the value of h_{min} must be modified if h_r exceeds the current value of h_{min}, and the cell dimensions h_{cell}xw_{cell} should match h_rxh_r if h_r is higher than both h_{min} and h_{cell}.

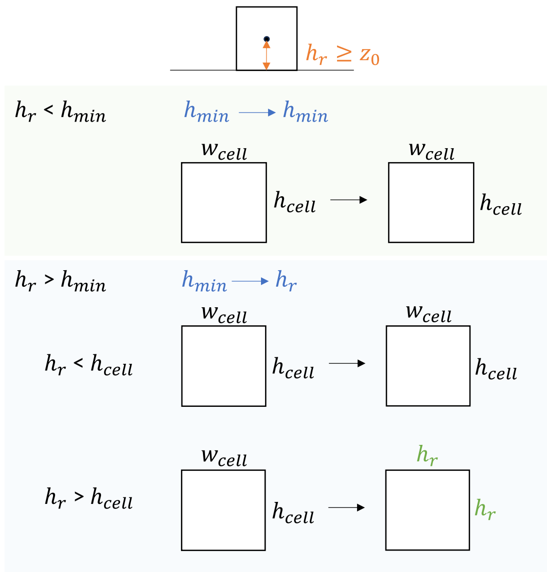


Figure 3.15.: Adjustment of cell dimensions based on the roughness height (h_{cell}xw_{cell}: cell dimensions, h_r: minimum cell height based on CFD guidelines related to the roughness height, and h_{min}: stored minimum cell height).

Algorithm 3.12: Roughness height (2 refinement boxes).

```
Data: roughness height $z_0$, cell dimensions $h_{cell}$x$w_{cell}$, and minimum cell height $h_{min}$
Result: cell dimensions $h_{cell}$x$w_{cell}$, and minimum cell height $h_{min}$
1 $h_r \leftarrow z_0 \cdot 2 \cdot 8$;
2 if $h_r > h_{min}$ then
3     $h_{min} \leftarrow h_r$;
4     if $h_{min} > h_{cell}$ then
5         $h_{cell} \leftarrow h_{min}$;
6         $w_{cell} \leftarrow h_{cell}$;
7         redefine the computational domain;
8         redefine the refinement boxes;
```

3.3.7. Maximum number of cells

An important limitation is the available memory of computers (Franke and Baklanov, 2007). Therefore, the prototype allows users to limit the total number of cells of the mesh N by indicating a threshold N_{max} (default value: $30 \cdot 10^6$ cells). Note that the mesh quality might not be guaranteed if this number is too low for the computational domain/geometry in question.

Algorithm 3.13 ensures that N_{max} is not exceeded by increasing the cell dimensions until N is lower than or equal to N_{max}. However, when N is significantly lower than N_{max}, the algorithm attempts to maximize the resolution by reducing the cell dimensions.

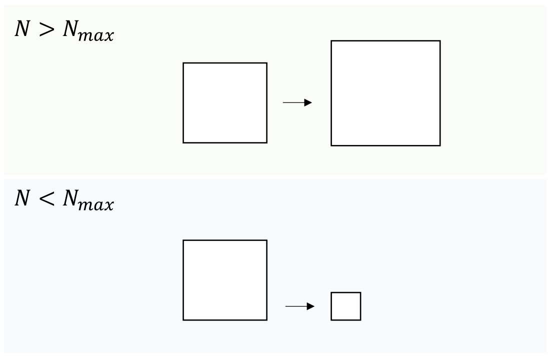


Figure 3.16.: Adjustment of cell dimensions based on the maximum number of cells N_{max} given by users.

Algorithm 3.13: Maximum number of cells.

```
Data: total number of cells $N$, threshold $N_{max}$, cell dimensions $h_{cell}$x$w_{cell}$, and
      minimum height $h_{min}$
Result: cell dimensions $h_{cell}$x$w_{cell}$
1  $reduction \leftarrow 0$;
2  $increase \leftarrow 0$;
3  while $N > N_{max}$ and $increase = 0$ do
4      $reduction \leftarrow 0$;
5      $h_{cell} \leftarrow h_{cell} + 0.1$;
6      $w_{cell} \leftarrow w_{cell} + 0.1$;
7      redefine the computational domain;
8      redefine the refinement boxes;
9      redefine $N$;
10 while $N < N_{max}$ and $h_{cell} \geq h_{min}$ and $reduction = 0$ do
11     $increase \leftarrow 1$;
12     $h_{cell} \leftarrow h_{cell} - 0.1$;
13     $w_{cell} \leftarrow w_{cell} - 0.1$;
14     redefine the computational domain;
15     redefine the refinement boxes;
16     redefine $N$;
17 if $h_{cell} < h_{min}$ OR $N > N_{max}$ then
18     $h_{cell} \leftarrow h_{cell} + 0.1$;
19     $w_{cell} \leftarrow w_{cell} + 0.1$;
20     redefine the computational domain;
21     redefine the refinement boxes;
22     redefine $N$ ;
```

To estimate the number of cells in a mesh, the following steps are followed for 3D models without terrain. Here a mesh with two refinement grids is assumed.

1. Compute the number cells around buildings.
   a) Compute the area of each building face.
   b) Determine the number of cells with refinement level 3 per area.
   c) Multiply this number by 4, as there are 4 rows needed to transition from one refinement level to the next.
   d) Repeat steps 1b and 1c with cells refined at level 2.
2. Compute the number of cells within refinement box 1 (excluding cells around buildings).
   a) Compute the volume of refinement box 1 (without building volume and the total cell volume occupied by cells around buildings).
   b) Divide this volume by cell volume with refinement level 2 to determine the number of cells within this region.
3. Compute the number of cells within refinement box 2.
   a) Compute the volume of refinement box 2 (without refinement box 1).

    b) Divide this volume by cell volume with refinement level 1 to find the number of cells within this region.

4. Compute the number of cells within the remaining area of the domain
    a) Compute the volume of the remaining area of the domain (without refinement boxes 1 and 2).
    b) Divide this volume by cell volume with refinement level 0 to find the number of cells within this area.
5. Sum up all number of cells to approximate the total number of cells in the mesh.

When 3D models include terrain surfaces, this number of cells is approximated by performing the following computations.

1. Perform the same computations as in step 1 of the number of cells approximation for models without terrain.
2. Compute the volume beneath the terrain surfaces within refinement box 1, refinement box 2, and the remaining area of the domain.
3. Compute the volume of refinement box 1 and subtract the volume occupied by the terrain (volume under terrain surfaces).
4. Repeat step 3 for refinement box 2.
5. Repeat step 3 for the remaining area of the domain.
6. Divide these volumes by the corresponding cell volume (cell volume with refinement level 1 in box 2, refinement level 1 in box 2, and refinement level 0 in the remaining area of the domain) to find the number of cells in each region.
7. Sum up all these determined number of cells to estimate the total number of cells in the mesh.

3.3.8. At least 10 cells per cube root of the building volume

At least 10 cells per cube root building volume is recommended. Algorithms 3.14 and 3.15 check for each building whether this requirement is met. This verification consists of computing the cube root of the building volume, dividing it by the width of the cell with the highest refinement level, and comparing it to 10. If the result is greater than 10, the building is valid; otherwise, it is considered invalid. If not all buildings meet this requirement, the cell width is iteratively reduced. The verification is performed again at each iteration. The iterations stop if all buildings respect this guideline, the cell ratio becomes lower than a specific threshold r_{min}, or the maximum of cells N_{max} is reached. This cell ratio should be approximately 1, as explained in Section 2.1.3. Assuming that the cell height is 1m, a cell ratio of 1.20 corresponds to a cell width of 0.80m. After each cell width reduction, the domain and refinement box dimensions were adjusted to ensure that a whole number of cells fit into them.

Users are provided with output data showing the buildings that do and do not respect the guideline mentioned above. First, the number of valid buildings is computed. Second, two OBJ files are shared: one containing valid buildings and one with invalid buildings. These output files are generated to allow users to adjust their cell ratio threshold r_{min} and cell refinement levels.

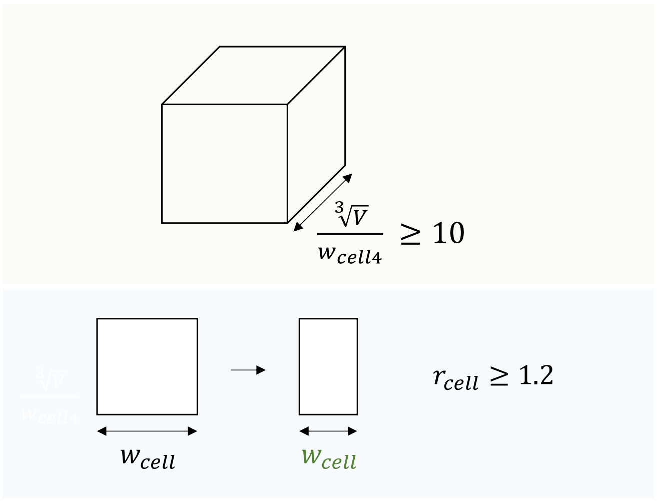


Figure 3.17.: Adjustment of cell dimensions to fit at least 10 cells per cube root of the building volume.

Algorithm 3.14: Cells per cube root of the building volume I.

```
Data: buildings $B$
Result: pair with valid buildings $s_n$ in %
1  $N \leftarrow$ total number of buildings;
2  $N_{valid} \leftarrow$ total valid buildings;
3  for all $b \in B$ do
4      $v \leftarrow$ volume of $b$;
5      if $v \geq 0$ then
6          add $v$ to $V$;
7          $cubeRoot \leftarrow \sqrt[3]{v}$;
8          $w_{cell3} \leftarrow$ cell width at refinement level 3;
9          if $cubeRoot/w_{cell3} \geq 10$ AND $cubeRoot/h_{cell3} \geq 10$ then
10             add 1 to $N_{valid}$;
11 $s_n \leftarrow N_{valid}/N$;
12 return $s_n$;
```

Algorithm 3.15: Cells per cube root of the building volume II.

```
Data: minimum ratio $r_{min}$
Result: cell dimensions $h_{cell}$x$w_{cell}$, and pair with valid buildings $s_n$ in %
s ← output Algorithm 3.14;
r ← cell ratio $h_{cell}/w_{cell}$;
while $s_n < 1$ AND $r \leq r_{min}$ and $N \leq N_{max}$ do
    $w_{cell} \leftarrow w_{cell} - 0.1$;
    r ← redefine cell ratio;
    redefine the computational domain;
    redefine the refinement boxes;
    redefine s;
if $N > N_{max}$ OR $r > r_{min}$ then
    $w_{cell} \leftarrow w_{cell} + 0.1$;
    r ← redefine cell ratio;
    redefine the computational domain;
    redefine the refinement boxes;
    redefine s;
return $s_n$;
```

3.3.9. At least 10 cells per building separation

A building separation is considered as the distance between two buildings and should measure at least 10 cells. Algorithms 3.16 and 3.17 are developed to verify this requirement. However, this minimum might lead to excessive mesh complexity as some cities might have narrow streets. Therefore, these algorithms exclude separations below a specific distance $d_{separation}$. The default value of this threshold is 2m, but could also be user-defined. To help users determine an appropriate threshold, a histogram showing the street width distribution is displayed. Connected buildings are also excluded from verification because the separations are non-existent.

Figure 3.18 illustrates the cell distribution within the urban area. It shows that each building is surrounded by 4 rows of cells with refinement level 3 and the remaining cells have refinement level 2. This means that, to satisfy the guideline mentioned above, there must be at least 8 cells with refinement level 3 and 2 cells with refinement level 2 within $d_{separation}$. Algorithm 3.17 verifies this requirement for each building by using the distances between its ground surface vertices and those of the other buildings. If this condition is not met for all buildings, the cell width w_{cell} is iteratively reduced by steps of 0.1. The verification is performed again at each iteration. The iterations stop if all separations respect this guideline, the cell ratio gets lower than a specific threshold r_{min} (user-defined or default value 1.20), or the maximum of cells N_{max} is reached. After each cell reduction, the domain and refinement box dimensions are adjusted to ensure that an integer number of cells fit into them.

Users are informed on the valid and invalid buildings with regard to this building separation guideline. First, the number of valid buildings is provided. Second, two 3D models are generated: one showing the valid building separations, and another containing the invalid ones. In addition, a text file is supplied including details on the invalid separations, such as vertex coordinates, distance, and maximum number of cells that can fit. These output files are generated to allow users to adjust their cell ratio threshold r_{min} and cell refinement levels.

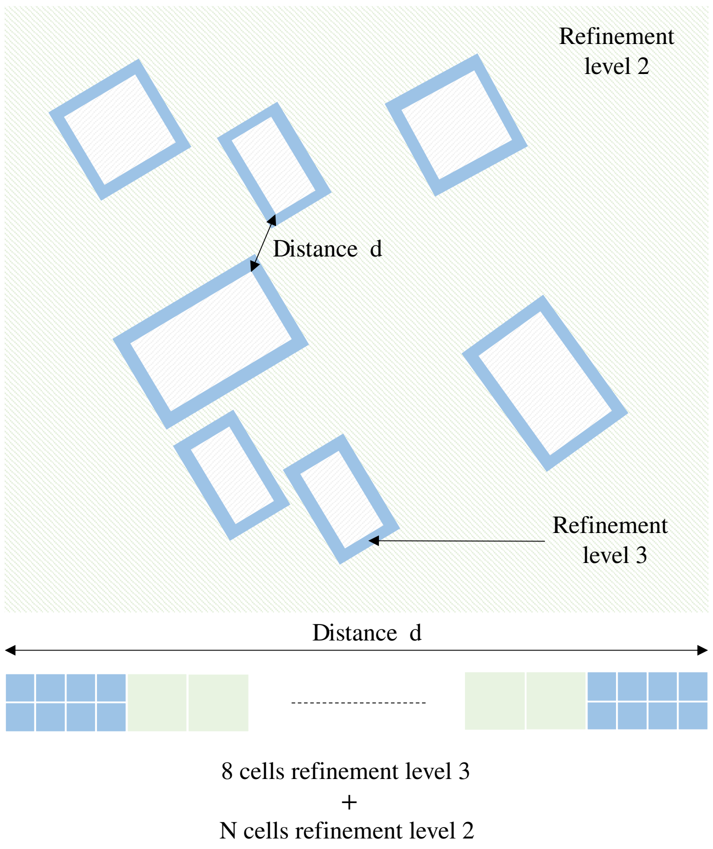


Figure 3.18.: Number of cells per building separation.

Algorithm 3.16: Cells per building separation I.

```
Data: buildings $B$, and distance threshold $d_{separation}$
Result: percentage of valid buildings $s_n$ in %
$w_{cell3} \leftarrow$ cell width at refinement level 3;
$w_{cell2} \leftarrow$ cell width at refinement level 2;
$min \leftarrow 2 \cdot 4 \cdot w_{cell3} + 2 \cdot w_{cell2}$;
$N \leftarrow$ total number of buildings;
$N_{valid} \leftarrow$ total valid buildings;
compute distances between buildings;
for all $b \in B$ do
    $v \leftarrow$ volume of $b$;
    if $v \geq 0$ then
        $s_{invalid} \leftarrow$ invalid separations;
        for all $s \in s_b$ do
            if $s \neq -1$ and $s \leq min$ then
                add $s$ to $s_{invalid}$;
    map $s_{invalid}$ to $b$;
    if $s_{invalid}$ is empty then
        add 1 to $N_{valid}$;
        label $b$ valid;
    else
        label $b$ invalid;
$s_n \leftarrow N_{valid}/N$;
```

Algorithm 3.17: Cells per building separation II.

```
Data: minimum ratio $r_{min}$
Result: cell dimensions $h_{cell}$x$w_{cell}$, and valid buildings $s_n$ in %
$s \leftarrow$ output Algorithm 3.16;
$r \leftarrow$ cell ratio $h_{cell}/w_{cell}$;
while $N \leq N_{max}$ and $s_n < 1$ and $r \geq r_{min}$ do
    $w_{cell} \leftarrow w_{cell} - 0.1$;
    $r \leftarrow$ redefine cell ratio;
    redefine the computational domain;
    redefine the refinement boxes;
    redefine $s$;
    redefine $N$;
if $N > N_{max}$ or $r > r_{min}$ then
    $w_{cell} \leftarrow w_{cell} + 0.1$;
    $r \leftarrow$ redefine cell ratio;
    redefine the computational domain;
    redefine the refinement boxes;
    redefine $s$;
    redefine $N$;
```

3.3.10. Region of Interest (RoI)

As explained in Section 2.2.3, the prototype focuses on the Region of Interest (RoI) introduced by Liu et al. (2018) and described as a multiplier of the largest building dimension L around the target buildings. As they suggest to include buildings within a radius of at least $3{\cdot}L$ around the target building, the prototype applies this recommendation.

Algorithm 3.20 defines the RoI. Users have the option to specify a target building by indicating one of the vertices from its ground surfaces. In this case, the algorithm defines the centre of the RoI as the centroid of this target building. When no target building is specified, its centre is set to the centre of the 3D city model. At the end, the buildings intersecting with the defined RoI are stored.

The RoI is generated to indicate to users which buildings are within this area. Two OBJ files are provided to users: one with the buildings within this region, and one demonstrating the RoI with a cylindrical shape. Additionally, a TXT file is given listing buildings inside the RoI.

Algorithm 3.18: RoI definition without target building Ia.

```
    Data: 3D city model $citymodel$ with extremities ($cx_{min}$, $cy_{min}$, $cz_{min}$) and ($cx_{min}$,
          $cy_{max}$, $cz_{max}$)
    Result: Region of Interest $RoI$
1   $p_x \leftarrow \frac{cx_{min}+cx_{min}}{2}$;
2   $p_y \leftarrow \frac{cy_{min}+cy_{min}}{2}$;
3   $p \leftarrow$ Point ($p_x$, $p_y$);
4   $max \leftarrow$ maximum building dimension;
5   $r \leftarrow 3 \cdot max$;
6   $RoI \leftarrow$ define $RoI$ with $p$ and $r$;
```

Algorithm 3.19: RoI definition with target building Ib.

```
    Data: x coordinate target $x$, and y coordinate $y$
    Result: Region of Interest $RoI$
1   $p_x \leftarrow x$;
2   $p_y \leftarrow y$;
3   $p \leftarrow$ Point ($p_x$, $p_y$);
4   search to which building $b$ $p$ belongs;
5   $c \leftarrow$ centroid of building $b$;
6   $max \leftarrow$ maximum dimension of $b$;
7   $r \leftarrow 3 \cdot max$;
8   $RoI \leftarrow$ define $RoI$ with $c$ and $r$;
```

Algorithm 3.20: RoI definition II.

```
Data: Buildings $B$, target indicator $target$, x coordinate target $x$, and y coordinate $y$
Result: buildings in RoI $B_{RoI}$
1 $RoI \leftarrow$ Region of Interest defined by Algorithm 3.18 or 3.19 depending on $target$;
2 for $all\ b \in B$ do
3     $intersection \leftarrow$ boolean that indicates whether $b$ and $RoI$ intersect;
4 if $intersection = true$ then
5     add $b$ to
```

3.3.11. Ground refinement

At the end of the mesh parameters definition process, parameters for a ground refinement are defined. Therefore, Algorithm 3.21 determines the cell at which h_{user} is located and uses this value to compute the distance h from the ground at which the cells must be refined with the highest refinement level.

Algorithm 3.21: Ground refinement.

```
Data: target evaluation height $h_{user}$
Result: height of layer with cells refined at level 3 $h$, and cell in which $h_{user}$ is
        located $cell$
1  $h_{cell_3} \leftarrow$ cell height at refinement level 3;
2  if $h_{user}$ is at the $3^{rd}$ cell from the ground then
3      $h \leftarrow 3 \cdot h_{cell_3}$;
4      $cell \leftarrow 3$;
5  else if $h_{user}$ is at the $4^{th}$ cell from the ground then
6      $h \leftarrow 4 \cdot h_{cell_3}$;
7      $cell \leftarrow 4$;
8  else if $h_{user}$ is at the $1^{st}$ or $2^{nd}$ cell from the ground then
9      $h \leftarrow 3 \cdot h_{cell_3}$;
10     $cell \leftarrow \frac{h_{user}}{h_{cell_3}}$;
```

As presented in Figure 3.7, two *snappyHexMesh* files with ground refinement are reported to users: one with cell limitation, and one without. For the last one, Algorithm 3.13 must be executed again, yet, with a different method to approximate the number of cells N.

To approximate the number of cells for a mesh with ground refinement, the following steps are followed (when two refinement boxes are applied). This method is developed for 3D models without terrain surfaces, as models with terrain have already ground refinement due to cells around geometries having the highest refinement level.

1. Compute the number of cells around buildings (as described in Section 3.3.7).
2. Compute the number of cells in refinement box 1.
   a) Compute the area occupied by cells around buildings.

i. Compute the perimeter of each building (approximated using the alpha-shape).

ii. Divide this perimeter by the width of cells with refinement level 3.

iii. Multiply the resulting number by 4 to obtain the number of cells around buildings in 2D.

iv. Multiply this number by the area of cells with refinement level 3.

b) Find the number of cells with refinement level 3 due to ground refinement.

i. Compute the area of refinement box 1 (without the building area and the area occupied by cells around buildings, computed in step 2e).

ii. Divide this area by the area of cells with refinement level 3 (w_{cell}xw_{cell}) and multiply the resulting number by the number of layers needed for the ground refinement.

c) Multiply the total number of cells by the corresponding cell volumes to find the volume occupied by these cells.

d) Subtract the resulting volume from the volume of refinement box 1 and divide this number by cells with refinement level 2.

3. Compute the number of cells in refinement box 2.

a) Compute the area of refinement box 2 (without refinement box 1).

b) Find the number of cells with refinement level 3 due to ground refinement, as described in 2b but then with refinement box 2.

c) Find the number of cells with refinement level 2 due to ground refinement: repeat the previous step but with refinement level 2 and a factor of 4 instead of the number of layers needed for the ground refinement (for the transition from one cell refinement to the other).

d) Compute the total volume occupied by cells due to ground refinement and subtract this volume from the volume of refinement box 2, and divide this number by cells with refinement level 1.

4. Compute the number of cells in the remaining area of the domain.

a) Repeat the three first steps as for the second refinement box, but replace ″refinement box 2″ by ″the remaining area of the domain″, and ″refinement box 1″ by ″refinement box 2″.

b) Determine the number of cells with refinement level 1 due to ground refinement.

c) Compute the total volume occupied by cells due to ground refinement and subtract this volume from the volume of this remaining area, and divide this number by cells with refinement level 0.

5. Sum up all the computed numbers of cells to approximate the total number of cells within the mesh with ground refinement.

3.3.12. Creating configuration files for OpenFOAM: blockMeshDict and snappyHexMeshDict files

As described in Section 2.1.3, the *blockMesh* and *snappyHexMesh* functions are used to generate meshes in OpenFOAM. Therefore, the *blockMeshDict* and *snappyHexMeshDict* files are needed. By using the domain dimensions, refinement box dimensions, cell sizes, and ground refinement settings defined with the developed workflow, they can be created.

In the *blockMeshDict* file (Appendix H), the final background mesh is defined by specifying the dimensions of the domain and the number of cells in each direction. The latter is computed by dividing one of the three domain dimension by the cell height h_{cell} or width w_{cell}, depending on the direction.

The *snappyHexMeshDict* file (Appendix I) contains information on the refinements within the mesh. First, it specifies the dimensions and refinements levels for each refinement box. Second, if a ground refinement is needed (Section 3.3.11), it should also be added. This is achieved by defining a plane with a vertex corresponding to the lowest corner of the domain and a normal vector (0, 0, 1).

4. Implementation

This chapter introduces the prototype in which the method, identifying geometric errors and defining mesh parameters, is implemented (Section 4.1). The chapter also presents the datasets used to test and validate the prototype (Section 4.2).

4.1. Prototype

The geometric validation and mesh parameters definition algorithms (backend) were written in C++ as it is fast and includes useful libraries. The Computational Geometry Algorithms Library (CGAL), which provides valuable tools for geometrical operations, was the most important library (`https://www.cgal.org/`).

The backend also includes programs for specific tasks. The val3dity tool (`https://github.com/tudelft3d/val3dity`) was used to verify individual building and terrain geometries. This program allows the validation of 3D primitives according to the international standard ISO19107, as described in Section 2.3.4. The CFD software OpenFOAM (`https://www.openfoam.com/`), introduced in Section 2.1.3, and the visualisation application ParaView (`https://www.paraview.org/`) were used to partially test the mesh parameters definition algorithms.

The user interface (UI) was created in Flask (`https://flask.palletsprojects.com/en/3.0.x/`), a web application framework written in Python. This framework was chosen due to its ability to build a UI in a relatively fast and easy way (the development of a UI was not the main objective of this thesis).

Figure 4.1 illustrates the pipeline of the prototype.

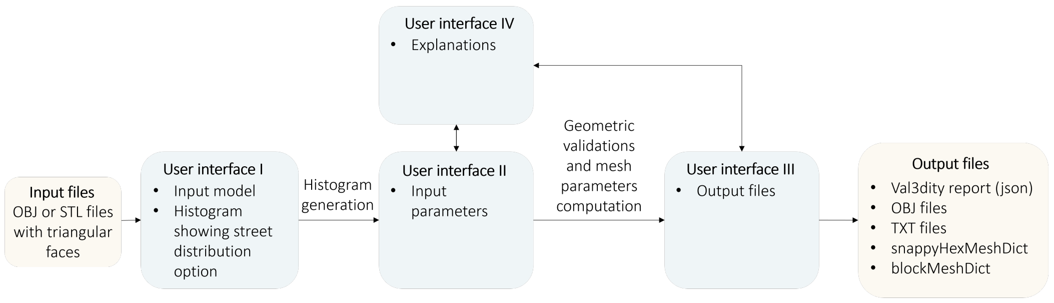
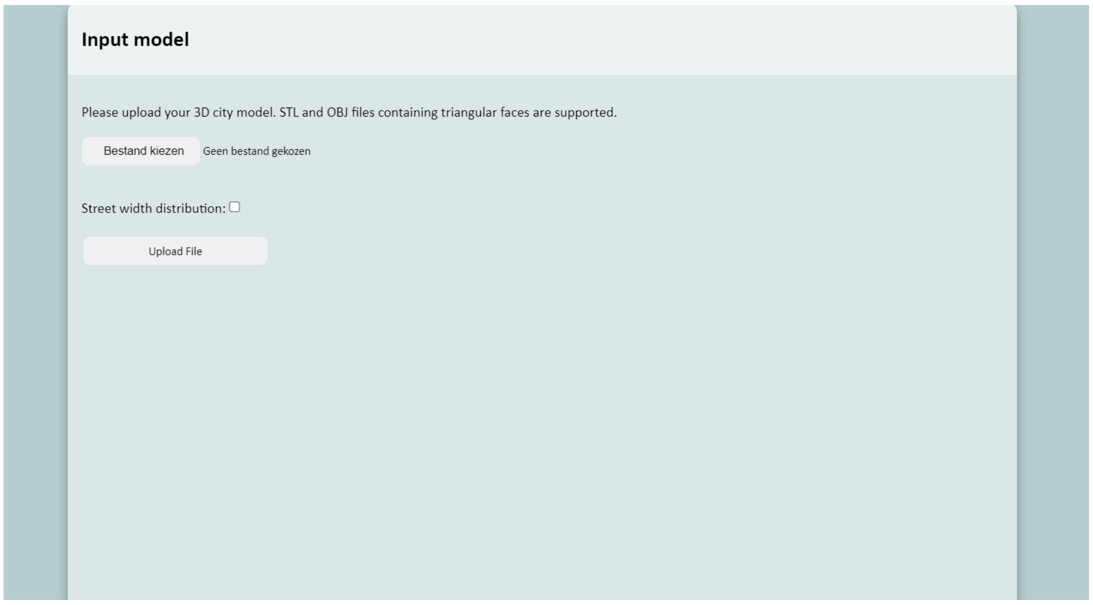


Figure 4.1.: Pipeline of the prototype.

User Interface I: Input file
Figure 4.2 shows the first page of the user interface, where users have to upload their 3D city model. In addition, they have the option to generate a histogram showing the distribution of street width/building separation. This histogram is provided to help users find an adequate threshold $d_{separation}$, as explained in Section 3.3.9.


Input model

Please upload your 3D city model. STL and OBJ files containing triangular faces are supported.

Street width distribution: ☐

Figure 4.2.: User Interface I: Input file.

User Interface II: Parameters and User Interface IV: Explanation
After uploading the input file, parameters for geometric validations and mesh parameter definition can be set. Explanations are provided for each parameter (Figure 4.4). Therefore, users can click on the parameters for which they need information. The prototype directs them to the corresponding explanation in user interface IV.

The following parameters have to be chosen by users. For most of them, the default values are already set. The maximum roughness height z_0 is the only parameter that is required to be inserted by users.

Mesh parameters definition:

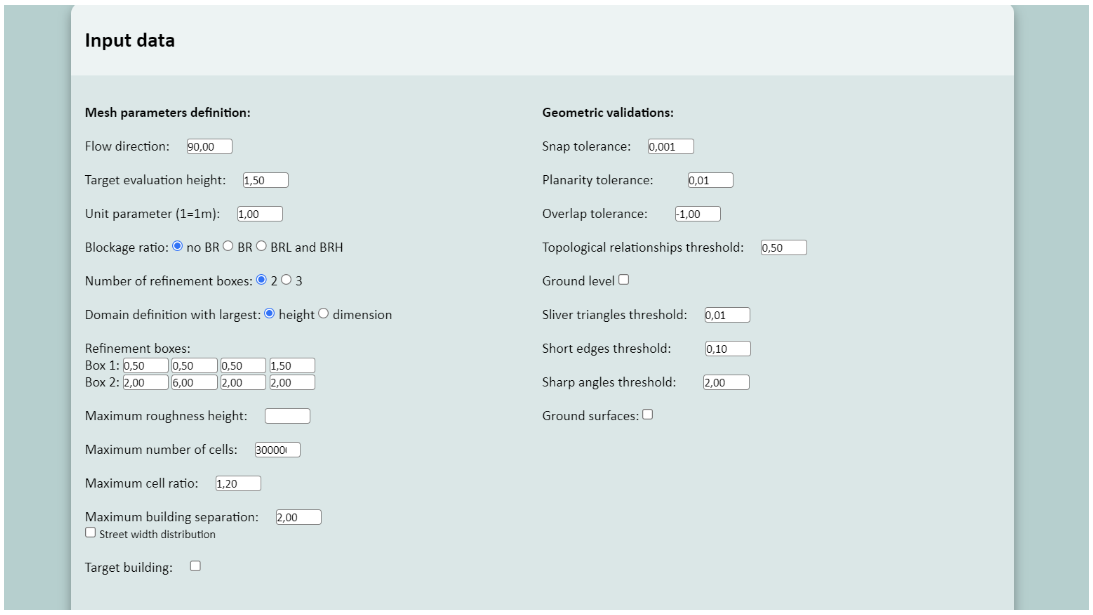
- Flow direction d_{flow},
- Target evaluation height h_{target},
- Unit parameter u,
- Blockage ratio BR,
- Number of refinement boxes N_{boxes},
- Dimension domain definition:
  - Height of the tallest building h_{max}, or

  - Largest building dimension d_{max}.
- Refinement boxes dimensions D_{box},
- Maximum roughness height z_0,
- Maximum cell ratio r_{min},

Geometric validations:

- Parameters needed for val3dity:
  - Snap tolerance,
  - Planarity tolerance,
  - Overlap tolerance.
- Topological relationships threshold ζ,
- Ground level z_t,
- Sliver triangles threshold θ,
- Short edges threshold λ,
- Sharp angles threshold θ,
- Parameters for ground surfaces definition:
  - Maximum height h,
  - Angle θ with regard to the height of the terrain.

More information on these parameters can be found in Appendix G providing the explanation given on User Interface IV.



Input data

Mesh parameters definition:

Flow direction: 90,00

Target evaluation height: 1,50

Unit parameter (1=1m): 1,00

Blockage ratio: [✓ no BR] BR BRL and BRH

Number of refinement boxes: [✓ 2] 3

Domain definition with largest: [✓ height] dimension

Refinement boxes:
Box 1: 0,50 0,50 0,50 1,50
Box 2: 2,00 6,00 2,00 2,00

Maximum roughness height:

Maximum number of cells: 30000(

Maximum cell ratio: 1,20

Maximum building separation: 2,00
☐ Street width distribution

Target building: ☐

Geometric validations:

Snap tolerance: 0,001

Planarity tolerance: 0,01

Overlap tolerance: -1,00

Topological relationships threshold: 0,50

Ground level ☐

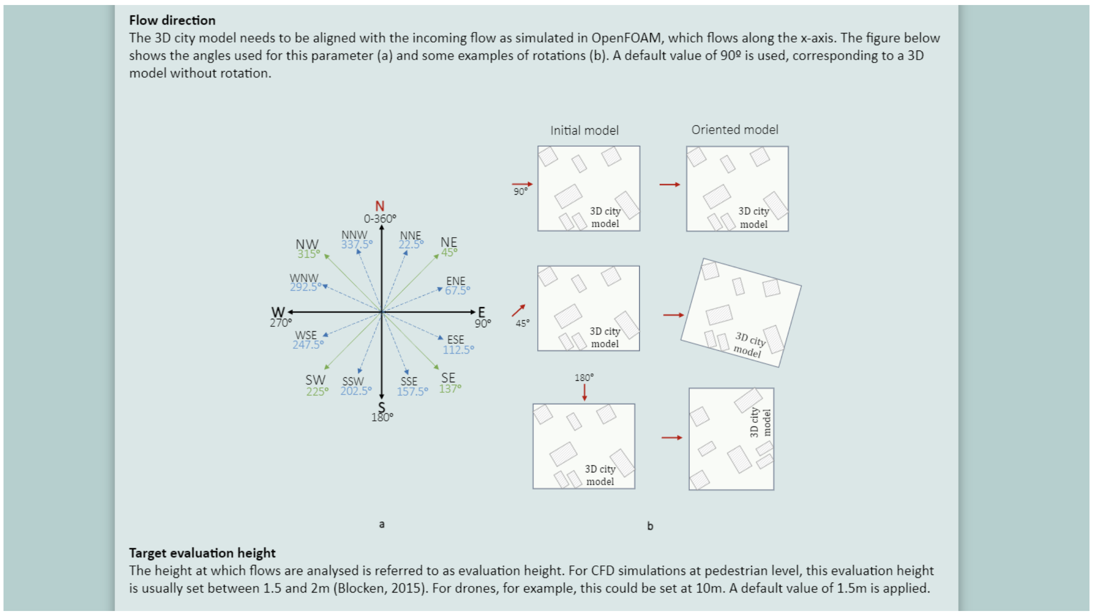
Sliver triangles threshold: 0,01

Short edges threshold: 0,10

Sharp angles threshold: 2,00

Ground surfaces: ☐

Figure 4.3.: User Interface II: Input parameters.

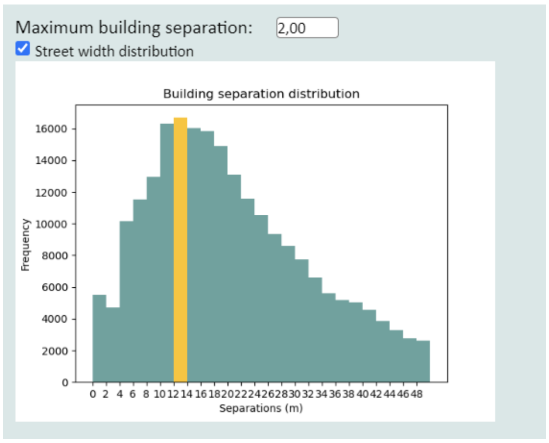


Flow direction
The 3D city model needs to be aligned with the incoming flow as simulated in OpenFOAM, which flows along the x-axis. The figure below shows the angles used for this parameter (a) and some examples of rotations (b). A default value of 90º is used, corresponding to a 3D model without rotation.

Target evaluation height
The height at which flows are analysed is referred to as evaluation height. For CFD simulations at pedestrian level, this evaluation height is usually set between 1.5 and 2m (Blocken, 2015). For drones, for example, this could be set at 10m. A default value of 1.5m is applied.

Figure 4.4.: User Interface IV: Explanation.

As mentioned earlier, there is an option in User Interface I to display a histogram with the street distribution to help users find the most appropriate separation threshold for buildings $d_{separation}$. Figure 4.5 illustrates an example of such a histogram. The bin with the highest number of street widths/building separations is shown in yellow.



Maximum building separation: 2,00
☑ Street width distribution

Figure 4.5.: User Interface II: Histogram showing the street width/building separation distribution. Bin with the highest frequency value is illustrated in yellow.

User Interface III: Results
After setting the parameters listed previously, the prototype identifies geometric errors and defines mesh parameters, and provides results to users (Figure 4.6). Each result is labeled by error, warning, or advice. Table 4.1 presents this classification.

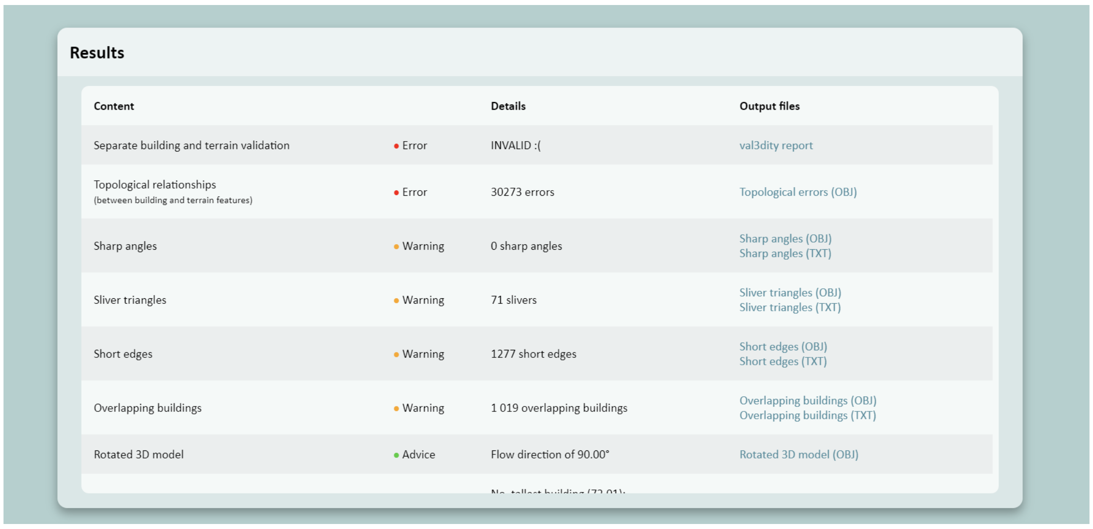
Table 4.1.: Classification of the results

Results	Class
Separate building and terrain validation (val3dity)	Indicated as error by the prototype, however, val3dity has its own classification.
Topological relationships between buildings and terrain	Error: floating building
	Warning: buildings too far below terrain surfaces or ground level
Sharp angles	Warning
Short edges	Warning
Sliver triangles	Warning
Overlapping buildings	Warning
Aligned 3D city model with incoming flow	Advice
Region of Interest (RoI)	Advice
Background mesh parameters (*blockMeshDict*)	Advice
Refined mesh parameters (*snappyHexMeshDict*)	Advice
Buildings satisfying required minimum of cells	Warning

The following output files are given to users:

- val3dity report (json), which describes errors and warnings from the separate building and terrain validations performed by the val3dity software.
- Topological errors, sharp angles, sliver triangles, overlapping buildings, and short edges in OBJ and TXT formats indicating their locations.
- Rotated 3D model, which is aligned with the incoming flow in OpenFOAM.
- *blockMeshDict* (Appendix H) and *snappyHexMeshDict* (Appendix I), which contain mesh parameters.
- Buildings and building separations with insufficient cells in OBJ and TXT formats, which indicates the locations of these buildings. In addition, the number of buildings and separations with sufficient cells is presented.
- Region of Interest (RoI) in OBJ and TXT formats, which indicates the locations of the buildings within the RoI.

More information on these output files can be found in Appendix G providing the explanation given on User Interface IV. .



Results

Content		Details	Output files
Separate building and terrain validation	● Error	INVALID :(	val3dity report
Topological relationships (between building and terrain features)	● Error	30273 errors	Topological errors (OBJ)
Sharp angles	● Warning	0 sharp angles	Sharp angles (OBJ) Sharp angles (TXT)
Sliver triangles	● Warning	71 slivers	Sliver triangles (OBJ) Sliver triangles (TXT)
Short edges	● Warning	1277 short edges	Short edges (OBJ) Short edges (TXT)
Overlapping buildings	● Warning	1 019 overlapping buildings	Overlapping buildings (OBJ) Overlapping buildings (TXT)
Rotated 3D model	● Advice	Flow direction of 90.00°	Rotated 3D model (OBJ)
		No. tallest building (72.01):	

Figure 4.6.: User Interface III: Results.

The code of the prototype can be found here: `https://github.com/MarenHengelmolen/thesis`.

4.2. Datasets

The 3D models used to test the prototype are listed below and illustrated in Figures 4.7 and 4.8. All datasets are in the Wavefront Object (OBJ) format, except for the TU Delft campus 1, which is in Stereolithography (STL) format. The reason why the focus lies on these formats is that the meshing tool *snappyHexMesh* from OpenFOAM is only compatible with them. The second to fifth datasets are retrieved from 3DBAG (`https://3dbag.nl/en/viewer`), which provides up-to-date 3D building models of the Netherlands.

Table 4.2.: Datasets.

3D model	Source	Format	Size	Buildings	Area (m^2)	Description
Simple shapes	Self-made	OBJ	1-70 kB	0-21	25-780 000	Cubic shape, rectangular shape, cone, simple roof model, two connected triangles, single building models with or without terrain, and cubes (21 identical cubes equally separated). All of them, except the last one, can vary in size.
Ouddorp	3DBAG	OBJ	0.25 MB	170	692 829	A (part of a) village situated on the coast, characterized by small buildings of approximately the same height. The tallest building is 9.40m.
Maastoren	3DBAG	OBJ	2.59 MB	675	1 130 656	The third largest building of the Netherlands (168.05m), located in Rotterdam, and its surrounding environment.
Centre of Delft	3DBAG	OBJ	2.6 MB	1 019	283 831	A main square with a church of 72.01m and surrounded by city blocks with tight alleys or separated by canals.
Begijnhofbuurt	3DBAG	OBJ	5.34 MB	714	692 829	This district in Amsterdam is one of the most densely built areas in the Netherlands. The highest building is 56.11m.
TU Delft campus 1	GEO5015[1]	STL	0.98 MB	103	522 802	Part of the TU Delft campus with the EWI building (89.42m).
TU Delft campus 2	City4CFD	OBJ	45 MB	72	17 640 000[2]	Part of the TU Delft campus with the EWI building (89.42m).

3D model	Source	Format	Size	Buildings	Area (m^2)	Description
Buildings	OpenFOAM	OBJ	12.9 MB	26	32 086	Small building model that represents a city and consists of mostly high buildings (the highest one is 76m).

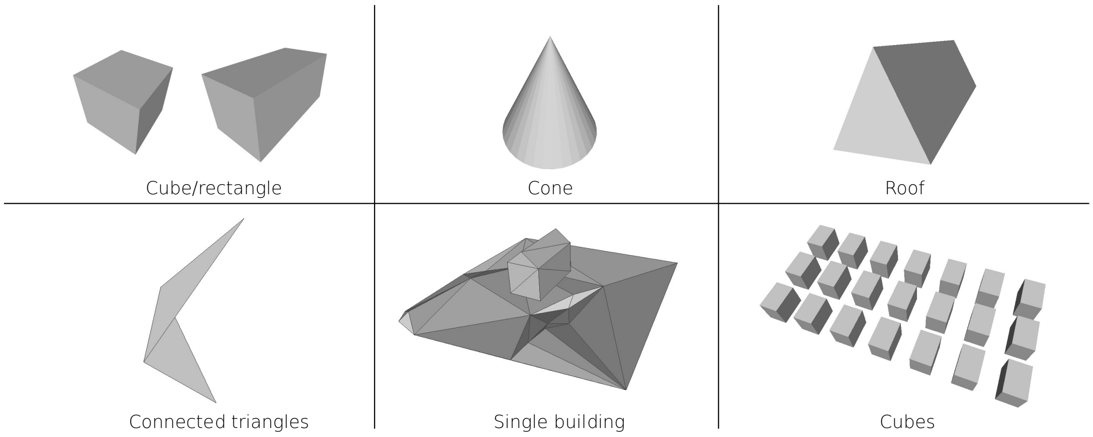


Figure 4.7.: Datasets: simple shapes.

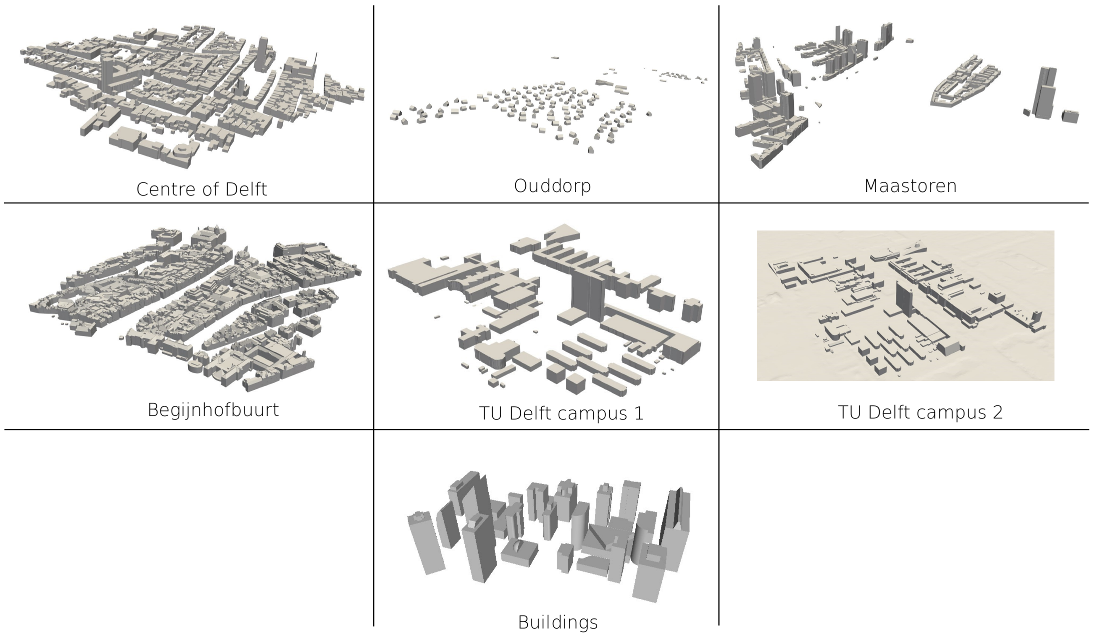


Figure 4.8.: Datasets: 3D city models.

[1]GEO5015: Modelling wind and dispersion in urban environments course. Note that, in contrast to OBJ, STL is not compatible with val3dity, however, the prototype uses a rotated version of the input file, which is in OBJ.
[2]Only buildings: 636 052 m^2

5. Results and analysis

In this chapter, results and analysis are presented. First, the chapter delves into the validations of the methods and algorithms used to identify geometric errors (Section 5.1). Then, the results of the mesh parameters definition method are discussed (Section 5.2).

5.1. Geometric validations

5.1.1. Topological relationships between buildings and terrain validation

Self-made simple shapes (Section 4.2) are developed to test the topological relationships validation method. These models consist of single buildings with or without terrain, and can be classified into the following categories.

- **Single buildings with horizontal floors and terrain**
  The height of the terrain varies. Figure 5.1 illustrates some test cases for which the threshold ζ remains the same and is set to 1. It shows that the method selects the terrain surfaces lower than the ground surfaces of the building, or the ones higher than a specific threshold from these surfaces.

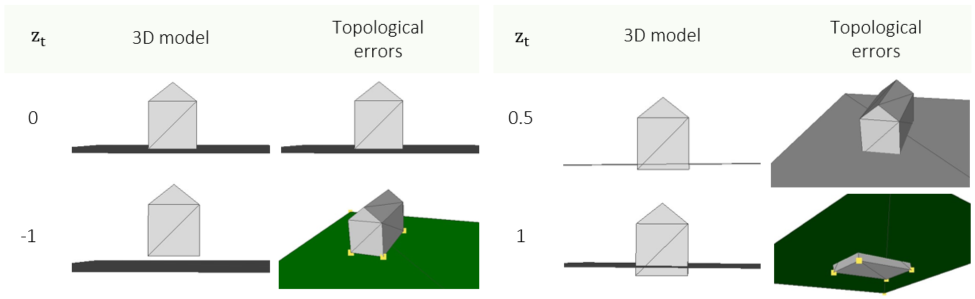


Figure 5.1.: Testing topological relationships validation with single building models, having horizontal floors with a height of 0, and horizontal terrain surfaces with a height of z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

- **Single building with horizontal floors and sloped terrain**
  The height of the terrain varies, but ζ remains the same. Figure 5.2 shows these models, and demonstrates that the algorithm identifies the terrain surfaces incorrectly placed with respect to the buildings. For example, in the second case, the terrain surface on

the right of the building remains unselected because its highest vertex is located at a distance less than ζ from the ground surface positioned above it.

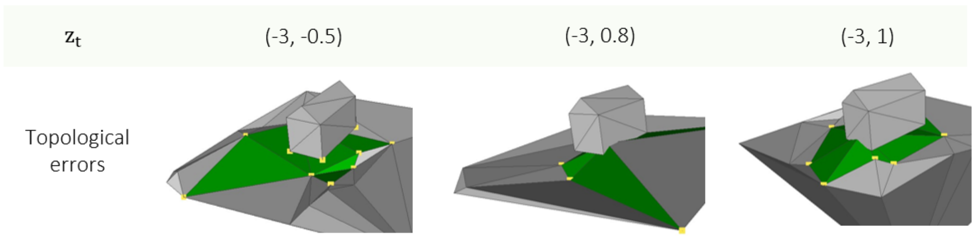


Figure 5.2.: Testing topological relationships validation with single building models, having horizontal floors with a height of 0, and terrain surfaces with variable heights within a range z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

- **Single building with sloped floors and terrains**
  The height of the terrain surfaces and building floors vary. Threshold ζ remains the same and is set to 1. Figure 5.3 demonstrates that buildings having their ground vertices partially placed on the terrain are considered invalid. In addition, it shows that buildings with floors slightly sloped can be considered as valid, as long as these ground surfaces are not higher or separated with a larger distance than ζ from the terrain.

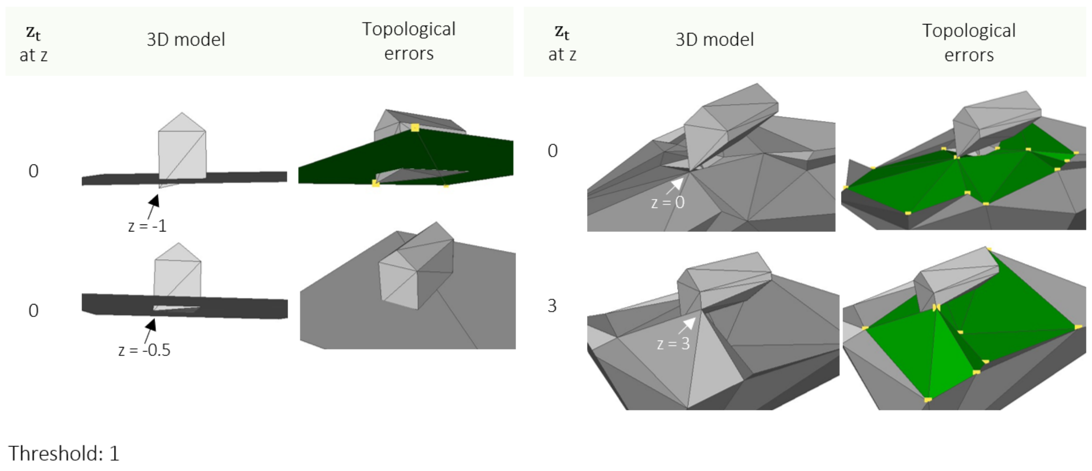


Figure 5.3.: Testing topological relationships validation with single building models, having floors with variable heights z, and terrain surfaces with variable heights z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

- **Single building without terrain**
  The height z_t at which the building must be placed varies, and the threshold ζ is set to

1. Figure 5.4 illustrates that the method indicates the building vertices that are higher than z_t, or separated with a distance higher than ζ.

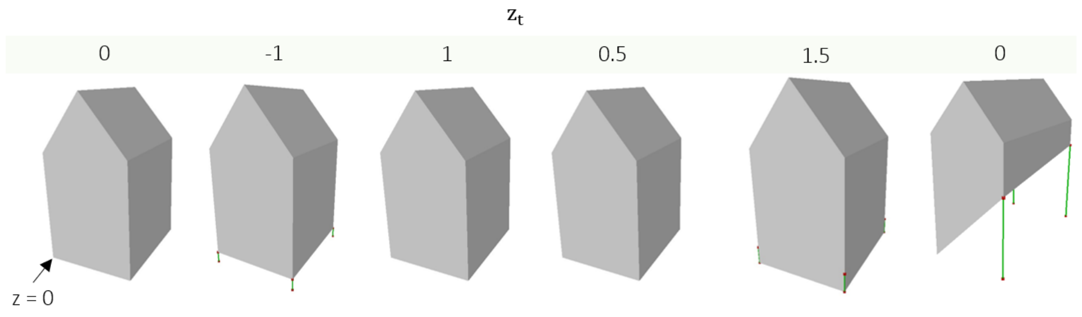


Figure 5.4.: Testing topological relationships validation with single buildings and without ground surfaces. z_t is the height at which buildings must be placed. The lowest z-value of the buildings (z_{min}) is 0. Threshold value ζ is set to 1.

More test cases can be found in Appendix A. The method used to identify topological relationships errors is based on ground surfaces (Section 3.2.2). So, it is also important that these ground surfaces are selected properly. Therefore, the algorithm selecting these surfaces was also tested. The results are presented in Appendix B.

The validation of topological relationships was also tested using 3D models representing existing areas in the city of Delft. First, the validation was performed with the centre of Delft, which does not contain terrain features. Initially, no ground level z_g was inserted; therefore, the method assumed that the buildings should be placed at the lowest z-value of the buildings z_{min}. The target height z_t is thus z_{min}. Figure 5.5 illustrates that almost all buildings were considered invalid, as indicated by the green edges at the bottom of them. This can be explained by the underground feature illustrated in Figure 5.6. With a ground level of 1 ($z_t = z_g = 1$), fewer buildings seem to be selected (Figure 5.7), and in contrast to the first scenario, the green edges indicating errors suggest that the buildings must be placed higher (Figure 5.8).

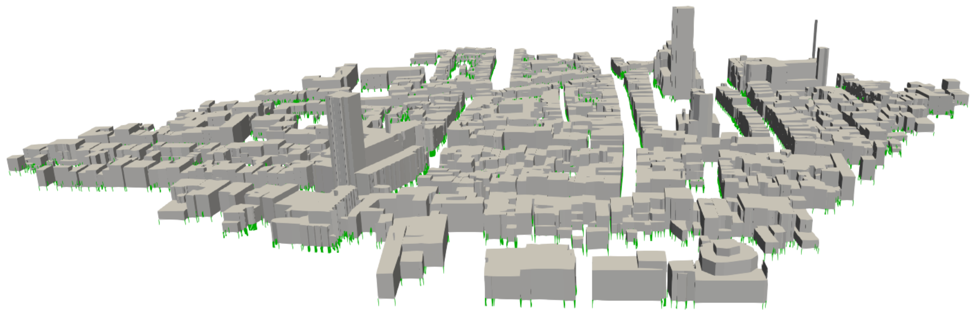


Figure 5.5.: Topological errors in the Centre of Delft indicated by green edges: buildings must be aligned with the lowest z-value of the urban model ($z_t = z_{min}$).

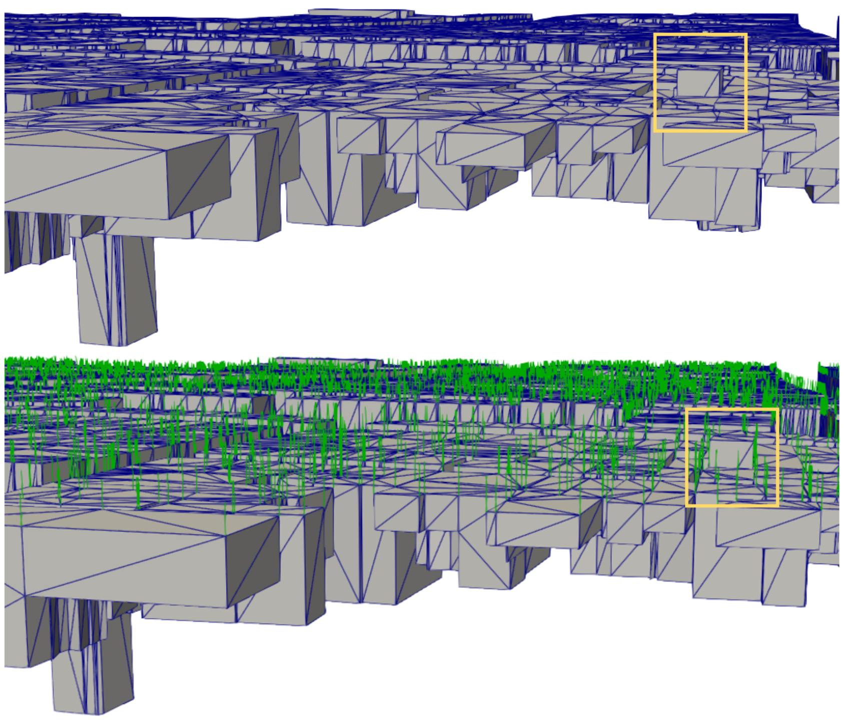


Figure 5.6.: Underground containing the lowest z-value of the Centre of Delft z_{min}.

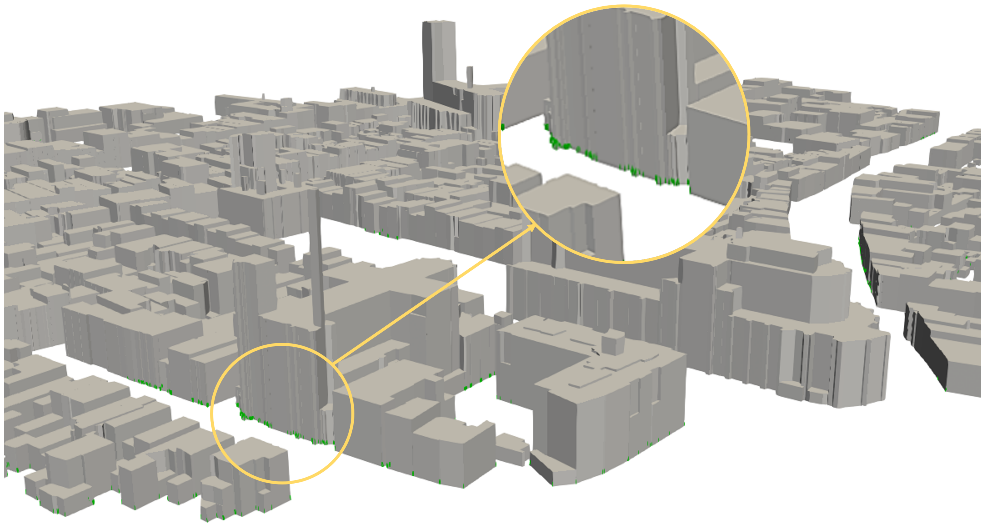


Figure 5.7.: Topological errors in the Centre of Delft indicated by green edges when a ground level of 1 is selected ($z_t = z_g$).

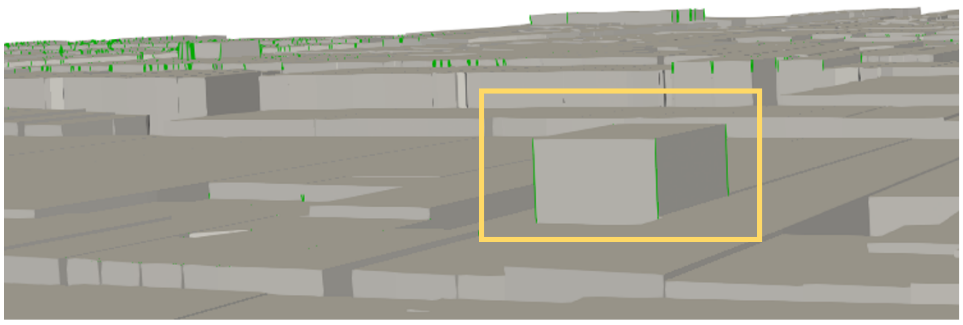


Figure 5.8.: Underground containing the lowest z-value z_{min} with the topological errors indicated by green edges when a ground level z_g of 1 is selected ($z_t = z_g = 1$).

Subsequently, the TU Delft campus file was used including terrain surfaces. As can be seen in Figure 5.9, almost all terrain surfaces placed under buildings were selected. This is probably due to small differences between the height of buildings and terrain objects. Yet, since space between buildings and terrain surfaces can drastically affect the CFD simulation results, these errors must be identified.

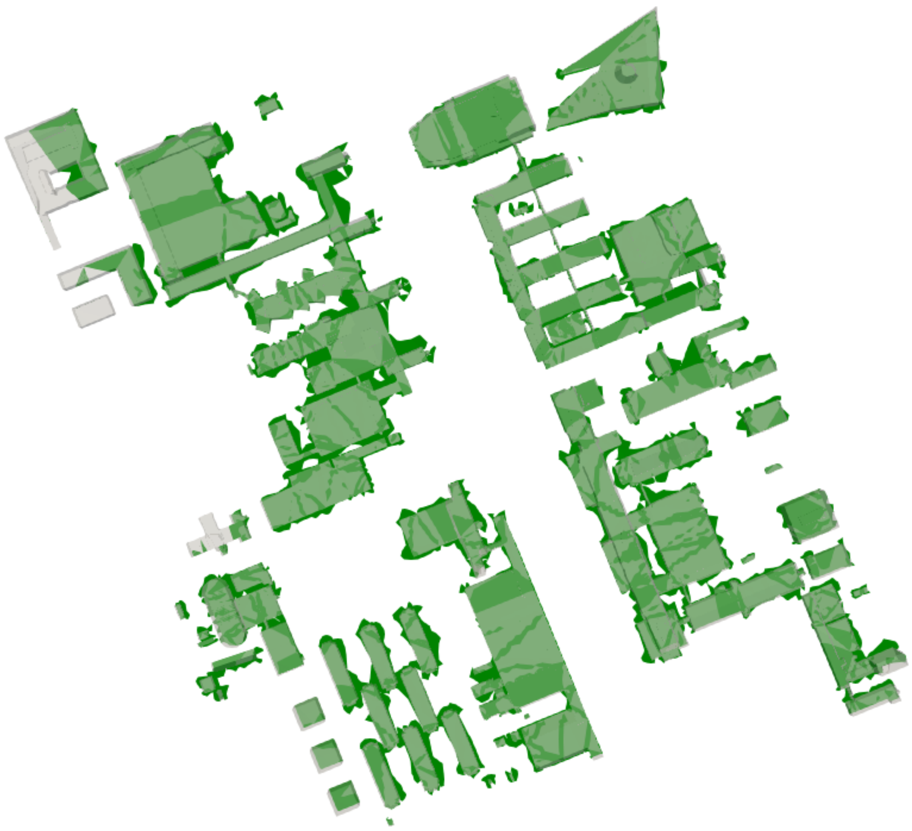


Figure 5.9.: Topological errors within the TU Delft campus model indicated in green.

The tests presented above demonstrate that topological errors were accurately identified with different building and terrain shapes, threshold values, and ground level values. Also, these errors seem to be correctly detected in real-world models. In short, based on the validations performed, this method appears to work properly.

5.1.2. Required validations for meshing in OpenFOAM

Sharp angles

To test whether the prototype is able to identify sharp angles, two connected triangles were modeled that are able to rotate around the axis passing through their common vertices (Figure 5.10). First, the sharp angles identification was performed several times with one of the triangles at 0° and the other one at different angles (first four triangles in Figure 5.11). The threshold θ remains the same for all tests. Second, both triangles were rotating and different thresholds were used (two last models in Figure 5.11). Third, a roof was modeled with different sizes to verify whether the algorithms was selecting all faces with small angles between them (Figure 5.12). Finally, two connected buildings were modeled to verify whether the algorithm identifies sharp angles between buildings.

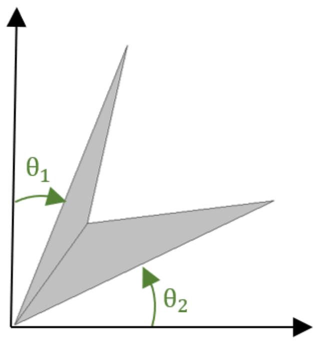
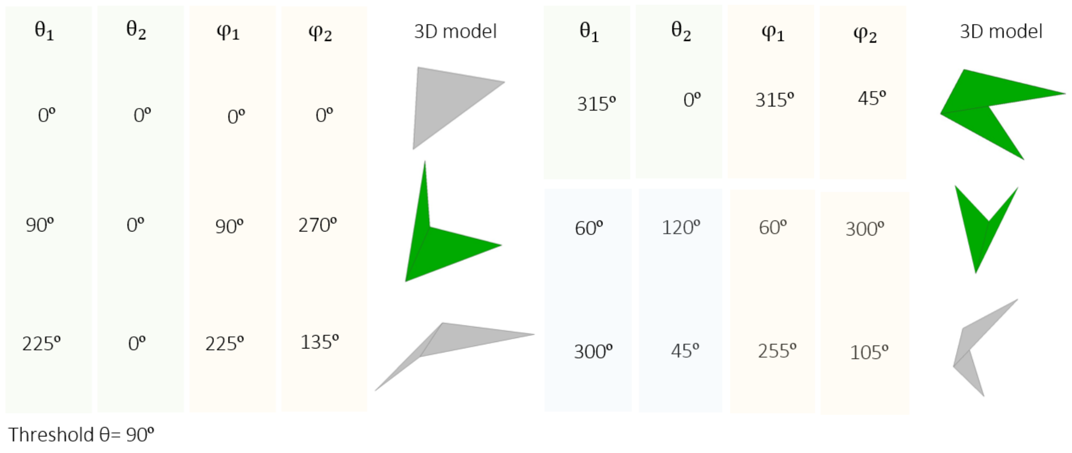


Figure 5.10.: Two connected triangles.



θ₁	θ₂	φ₁	φ₂	3D model	θ₁	θ₂	φ₁	φ₂	3D model
0º	0º	0º	0º		315º	0º	315º	45º	
90º	0º	90º	270º		60º	120º	60º	300º	
225º	0º	225º	135º		300º	45º	255º	105º	

Threshold θ= 90º

Figure 5.11.: Sharp angles identification of two connected triangles with a threshold value θ of 90°. θ_1 and θ_2 are the angles illustrated in Figure 5.10. ϕ_1 is the first angle between these triangles and ϕ_2 is the second one. Note that no sharp angle is identified with a threshold value of 0°, as neighbouring faces are defined as faces having two common and two uncommon vertices. Sharp angles are indicated in green.

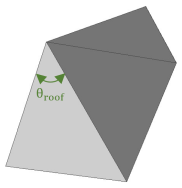


Figure 5.12.: Roof

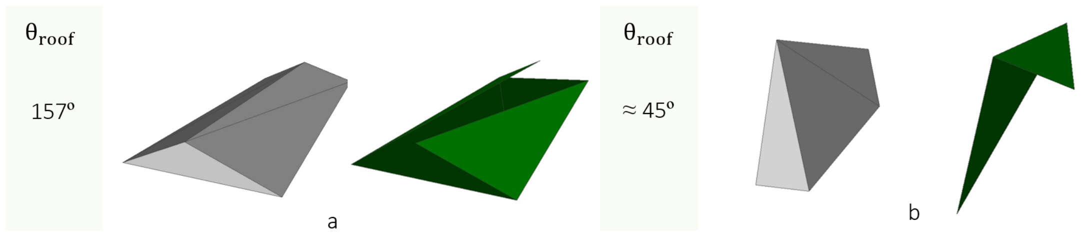


Figure 5.13.: Sharp angles identification with roofs and a threshold value θ of 90º. θ is the angle illustrated in Figure 5.12. Sharp angles are indicated in green.

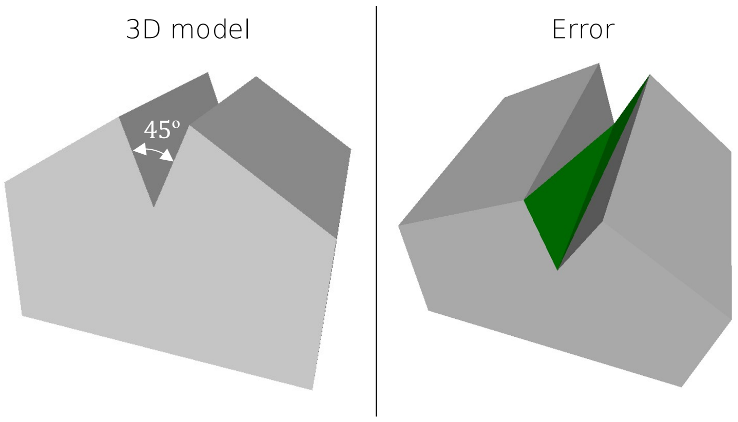


Figure 5.14.: Sharp angles identification between buildings with a threshold value θ of 45º. Sharp angles are indicated in green.

Figure 5.11 illustrates that the method correctly identifies sharp angles between two connected triangles. Additionally, it shows that both angles (ϕ_1 and ϕ_2) between these triangles are evaluated. This is demonstrated by the two first cases on the right in this figure: the first one has an angle ϕ_2 lower than the threshold value of 90º, and the second has an angle ϕ_2 lower than this value. Figure 5.13 shows that the method accurately detects multiple sharp angles in an object. For example, the faces at the bottom and sides of roof (a) were selected as the angles between them are smaller than threshold value of 90°. Figure 5.14 demonstrates that the algorithm correctly identifies sharp angles between buildings. In this figure, the connected faces from both roofs form an angle of 45º, which is correctly recognized as a sharp angle since the threshold was set to 90º.

Sliver triangles

The sliver triangles identification method was validated by using different models of cones. Each of them are divided into a certain number of triangles. The algorithm was executed with different thresholds. The results are summarised in Figure 5.16, and the algorithm seems to work properly as it selects the triangles having a sliver parameter lower than a specific threshold. Another validation method was to perform this validation with a roof with different sizes (Figure 5.17). It seems that the algorithm is able to select multiple triangles with different sliver parameters lower than a specific threshold.

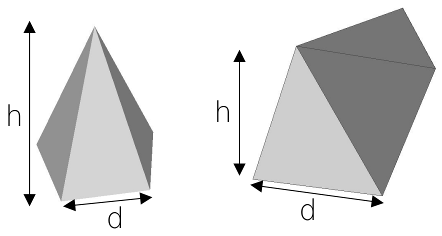


Figure 5.15.: Cone and roof models.

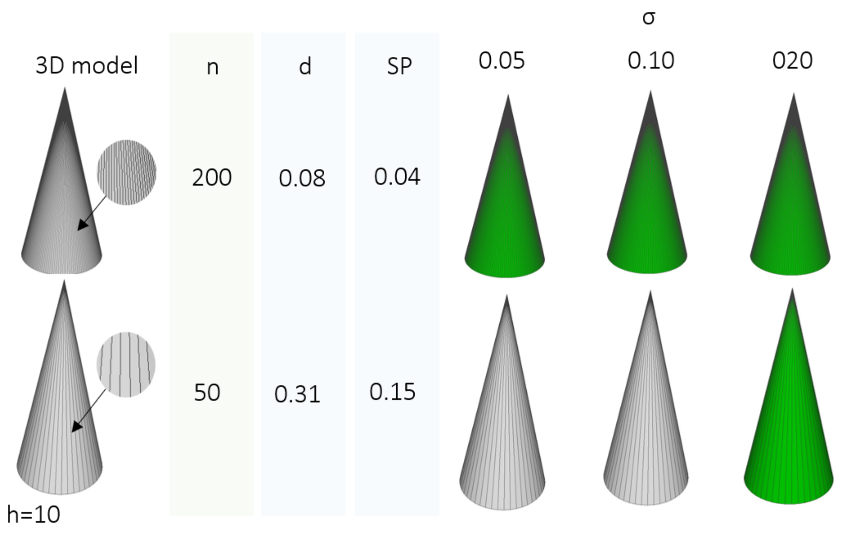


Figure 5.16.: Sliver triangles identification with cones (n: number of triangles, d and h: distances as illustrated in Figure 5.15, and SP: sliver parameter), and different threshold values σ for the sliver parameter. Sliver triangles are indicated in green.

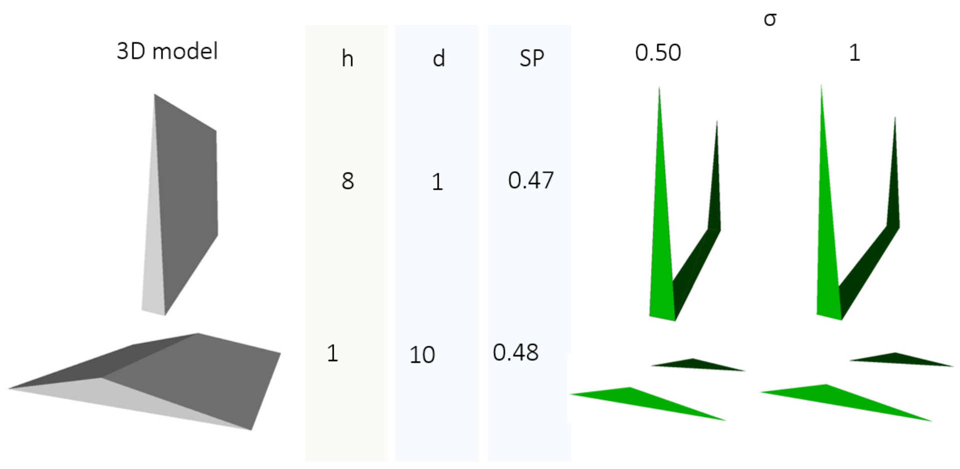


Figure 5.17.: Sliver triangles identification of roofs (n: number of triangles, d and h: distances as illustrated in Figure 5.15, and SP: sliver parameter), and different threshold values σ for the sliver parameter. Sliver triangles are indicated in green.

Short edges

To verify whether the short edges identification method, the same shape was used as for the sliver triangles (Section 5.1.2). Again, the cone was divided into a x number of triangles, and different threshold values λ were used during the validation. Figure 5.18 and 5.19 shows the results, and demonstrates that edges shorter than or equal to a threshold λ are selected. Thus, based on these validations, this short edges identification method seems to work properly.

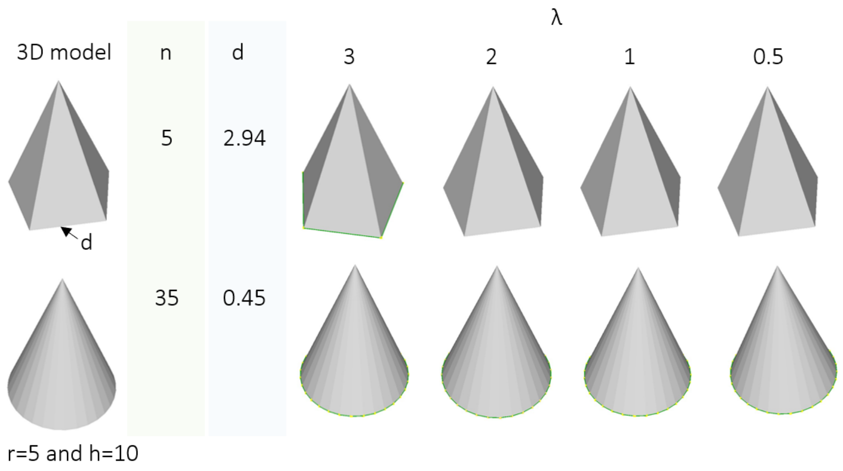


Figure 5.18.: Short edges identification of cones (n: number of triangles, d and h: distances as illustrated in Figure 5.15, and r: radius), and different threshold values for the length of edges. Short edges are indicated in green.

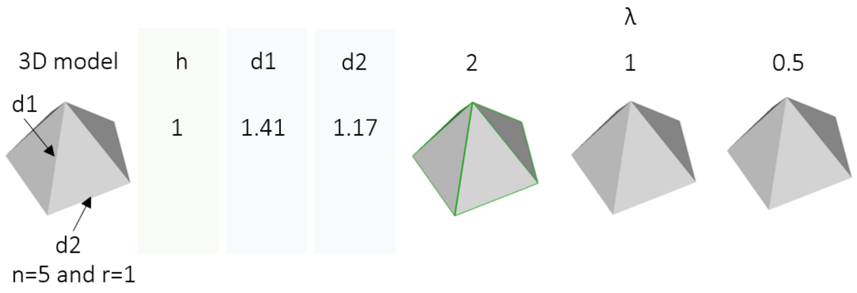


Figure 5.19.: Short edges identification of cones (n: number of triangles, h: distance as illustrated in Figure 5.15, and r: radius), and different threshold values for the length of edges. Short edges are indicated in green.

Overlapping buildings

To test Algorithm 3.5, pairs of buildings were modeled with different positions, including separated buildings, connected buildings, and overlapping buildings. Figure 5.20 shows these models and demonstrates that an error occurs only when buildings overlap. Note that intersections are allowed, as demonstrated by the connected buildings.

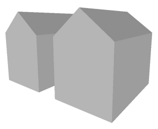
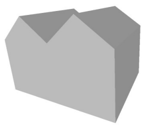
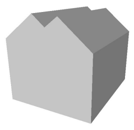
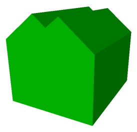
	3D model	Errors
Separated buildings		-
Connected buildings		-
Overlapped buildings		

Figure 5.20.: Overlapping buildings identification with two buildings. Buildings that are overlapping are indicated in green.

Real-world data

The algorithms that identify sliver triangles, sharp angles, and short edges were also tested using three different 3D city models representing the Begijnhofbuurt in Amsterdam, TU Delft campus, and Centre of Delft (Section 4.2). Figure 5.21, 5.22, and 5.23 show the results of these tests. For all of them, it can be observed that the number of errors identified increases with higher threshold values, which is as expected.

As described in Section 17, faces with collinear vertices are excluded during the identification of sliver triangles. Before filtering out these geometries and setting the threshold values to 0, the TU Delft campus file contained 52 sliver triangles. These selected geometries appeared to be mainly faces formed by collinear vertices. Since they are not necessarily sliver triangles and are already considered as errors by val3dity, the algorithm filters them out. Figure 5.22 demonstrates that this filtering task works properly, as no sliver triangles were selected when the threshold value was set to 0.

Unlike the algorithm for identifying sliver triangles, no errors were detected when the algorithm for identifying short edges was run with a threshold value of 0. However, it was noted that edges with identical vertices could also be selected as short edges, which may not be the most appropriate labeling and would already be considered as an error by val3dity. For this reason, an edge with identical vertices was added to the Centre of Delft file, which was indeed identified by the algorithm. After implementing the filter to exclude these edges, no short edges were selected, indicating that this filter works as expected.

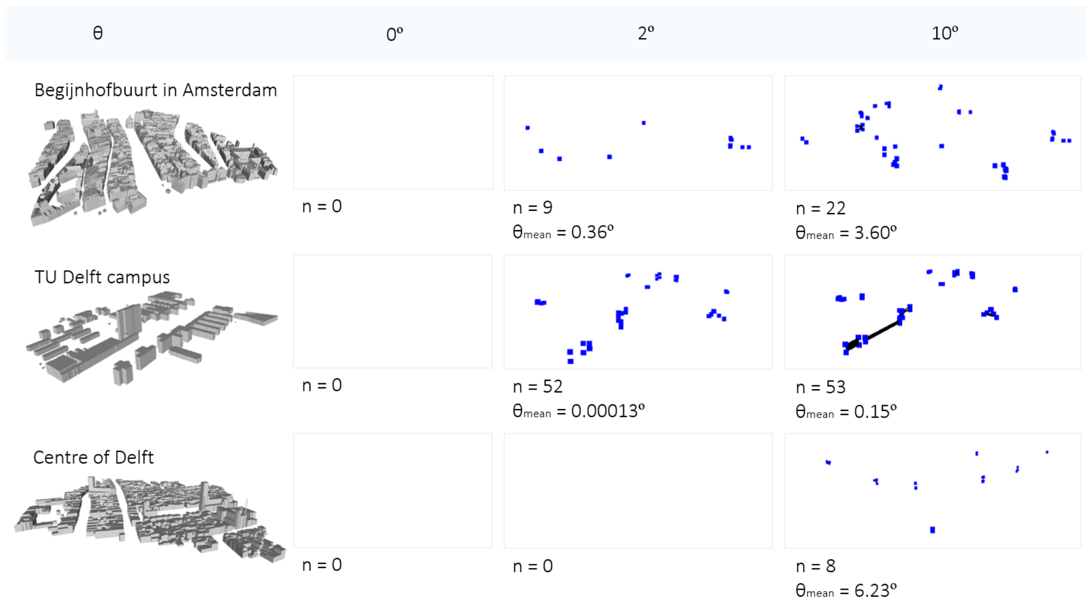


Figure 5.21.: Sharp angles identification with real-world data (θ: sharp angles threshold, n: number of sharp angles, and θ_{mean}: mean angle of sharp angles within urban model). Sharp angles are indicated in blue.

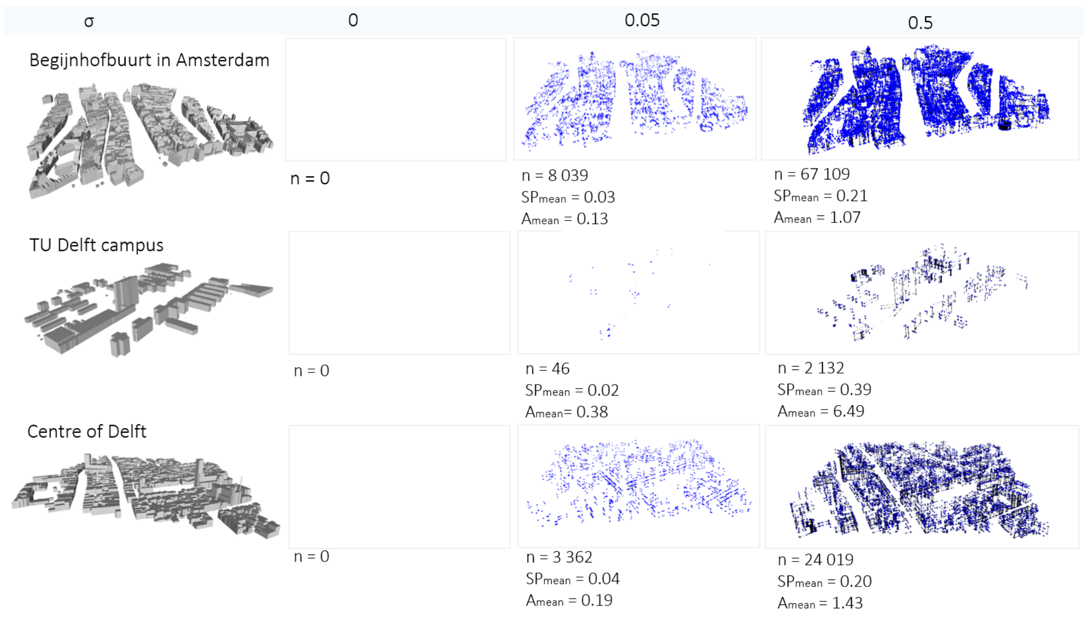


Figure 5.22.: Sliver triangles identification with real-world data. (σ: sliver parameter threshold (2xperimeter/area), n: number of sliver triangles, SP_{mean}: mean sliver parameter of sliver triangles within urban area, and A_{mean}: mean area of sliver triangles within urban area). Sliver triangles are indicated in blue.

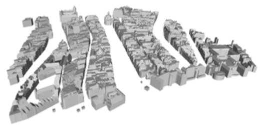
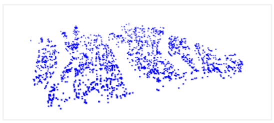
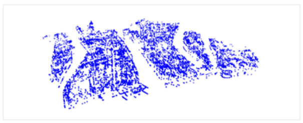
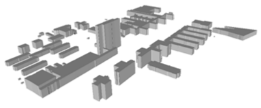
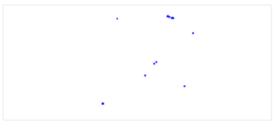
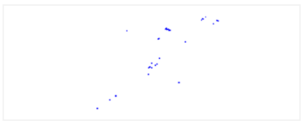
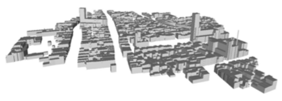
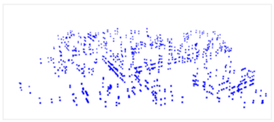
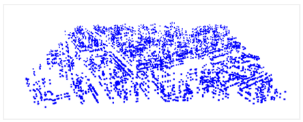
λ	0	0.10	0.5
Begijnhofbuurt in Amsterdam 	n = 0	 n = 3 461 l_{mean} = 0.06	 n = 30 938 l_{mean} = 0.26
TU Delft campus 	n = 0	 n = 15 l_{mean}= 0.04	 n = 36 l_{mean} = 0.14
Centre of Delft 	n = 0	 n = 1 277 l_{mean} = 0.07	 n = 12 451 l_{mean} = 0.25

Figure 5.23.: Short edges identification with real-world data (λ: short edges threshold, n: number of short edges, and l_{mean}: mean length of short edges within urban model). Short edges are indicated in blue.

The algorithm for identifying overlapping buildings was tested using the previously mentioned datasets and the one representing Ouddorp. Figure 5.24 presents the number of overlapping buildings for each dataset. The results indicate that test files with higher building density (Begijnhofbuurt and Centre of Delft) contain more overlapping buildings than the one with lower building density (Ouddorp). This could be explained by the complexity of modeling dense building models, which are more prone to geometric errors. The TU Delft campus file appears to be the only one without overlapping buildings, which is striking as overlapping buildings seem to common. Based on this qualitative assessment, the algorithm appears to work correctly.

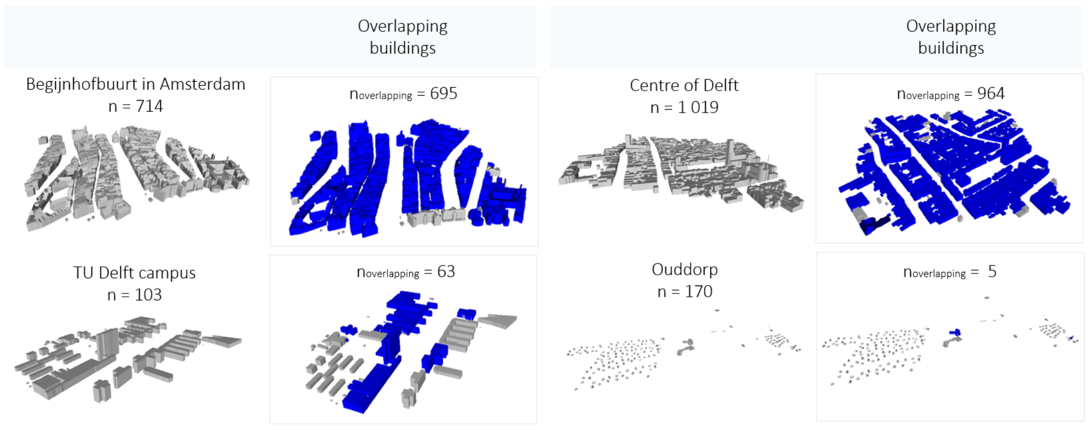


Figure 5.24.: Overlapping buildings identification with real-world data (n: number of buildings, and $n_{overlapping}$: number of overlapping buildings).

5.2. Preparation for CFD simulation steps

To evaluate the method to define mesh parameters presented in Section 3.3, several test cases were used. Each of them addresses a different sub-algorithm that is responsible for a single user-defined parameter or CFD guideline. Each model was then analysed on several parameters with a focus on cell dimensions suggested by the prototype. For the different parameters and guidelines from these sub-algorithms, the following question(s) were formulated and then explored for the individual test cases. The datasets (shown in Section 4.2) to analyse these parameters or guidelines are summarised in Table 5.1. The test cases mainly address meshes with two refinement boxes and models without terrain features.

- **Flow direction** d_{flow}: Does the 3D city models rotate around the z-axis in accordance with the flow directions illustrated in Figure 3.8? How does this orientation influence the cell dimensions defined by the prototype?
  - Cube: is the rotation correctly performed based on these flow directions?
  - Centre of Delft: is the orientation correct for all buildings?
  - TU Delft campus 2: is the orientation also correct for terrain features?
- **Computational domain**: Are the required minimum dimensions always followed? Are the blockage ratios BR, BR_H, and BR_L respected when applied? How does the height of the tallest building h_{max} influence the recommended cell dimensions?

  Three models with different h_{max} were selected:
  - Maastoren of Rotterdam: contains the third largest building of the Netherlands (h_{max} = 168.05m),
  - Centre of Delft: includes the "Nieuwe Kerk" of Delft (h_{max} = 72.01m),
  - Ouddorp: represents a village with small buildings of approximately the same height (h_{max} = 9.40m).
- **Evaluation height** h_{user}: When is the evaluation height respected?
  - Centre of Delft: contains a diverse range of building objects separated by various distances.
- **Roughness height** z_0: Is the roughness height always respected?
  - Centre of Delft: used for the same reasons as h_{user}.
- **Maximum number of cells** N_{max}: Is this limit always respected? How does it affect the final cell dimensions? Is the number of cells within the final mesh adequately estimated?
  - Centre of Delft: used for the same reasons as h_{user}.
- **At least 10 cells per cube root of the building volume**: Is this guideline correctly verified? Is the cell ratio r_{min} always respected? What is the impact on the cell dimensions?
  - Cubes: mainly to answer the first question as the volume of a cube can be easily verified.

  - Ouddorp and TU Delft campus 1: contain mainly small and large buildings, respectively. The difference in building dimensions makes it possible to observe to what extent this guideline is followed under different circumstances.

- **At least 10 cells per building separation**: Is the minimum number of cells per building separation correctly calculated? Is the cell ratio always respected? How does it influence the final cell dimensions?
  - Cubes: mainly to answer the first question as the model contains cubes separated with distances that can be readily verified.
  - The building distributions of Ouddorp and Begijnhofbuurt differ significantly. The first model contains buildings that are almost all separated with a relatively large distance; the second one represents the most densely built district of the Netherlands. The difference in building distribution makes it possible to observe to what extent this guideline is followed under different circumstances.
- **Region of Interest (RoI)**: is the RoI correctly defined?
  - Ouddorp: represents small buildings that are widely distributed, making it easier to verify whether the RoI is correctly defined.
- **Ground refinement**: what is the impact of the ground refinement on the final cell dimensions? How accurate is the approximation of the number of cells within the final mesh?
  - Centre of Delft: used for the same reasons as h_{user}.
- **Number of refinement boxes**: does the mesh parameters definition also work for meshes with three refinement boxes?
  - Centre of Delft: used for the same reasons as h_{user}.

Table 5.1.: Matrix showing 3D models per test case.

3D models	Cube	Cubes	Delft	Ouddorp	Maastoren	Begijnhofbuurt	TU Delft
Flow direction	■		■				
Computational domain			■	■	■		
Evaluation height			■				
Roughness height			■	■			
Max. number of cells			■	■			
Building volumes		■		■			■
Building separations		■		■		■	
RoI				■			
Ground refinement			■	■			
N. refinement boxes			■				

5.2.1. Flow direction d_{flow}

Figure 5.25 and 5.26 show that 3D models align with the incoming flow simulated in OpenFOAM, which flows along the x-axis. The first figure demonstrates the correct orientation with models without terrain features; the second one illustrates that all surfaces are rotated with a model with terrain surfaces, except for the water surfaces. This should be an easy fix in the algorithm, it is related to the semantics. The angles applied correspond to commonly used wind directions, as illustrated in Section 3.3.2. The flow direction does not seem to impact cell dimensions, but it does influence the domain dimensions (Table 5.2). This is due to the minimum and maximum x and y coordinates of the urban area that change with different angles.

Table 5.2.: Influence of the flow direction on the mesh suggested by the prototype for the Centre of Delft (N_{max}: 30 million cells, z_0: 0.5m).

d_{flow} (○)	0°	45°	157.5°	270°
hxw (m)	8x6.68	8x6.68	8x6.68	8x6.68
Domain volume (m^3)	$1.1 \cdot 10^9$	$1.3 \cdot 10^9$	$1.2 \cdot 10^9$	$1.1 \cdot 10^9$

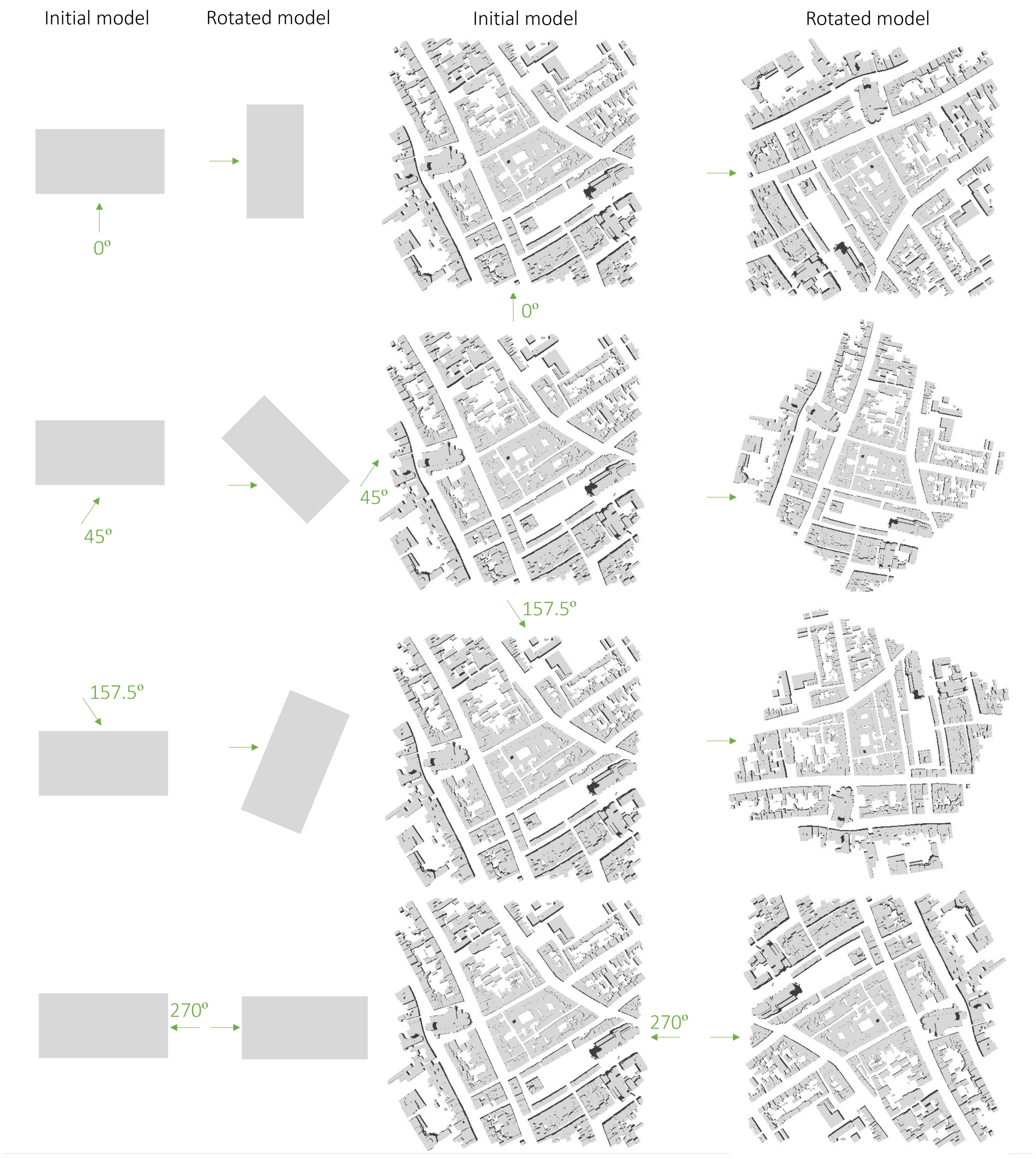


Figure 5.25.: Model rotation of a simple cube (left) and the Centre of Delft (right).

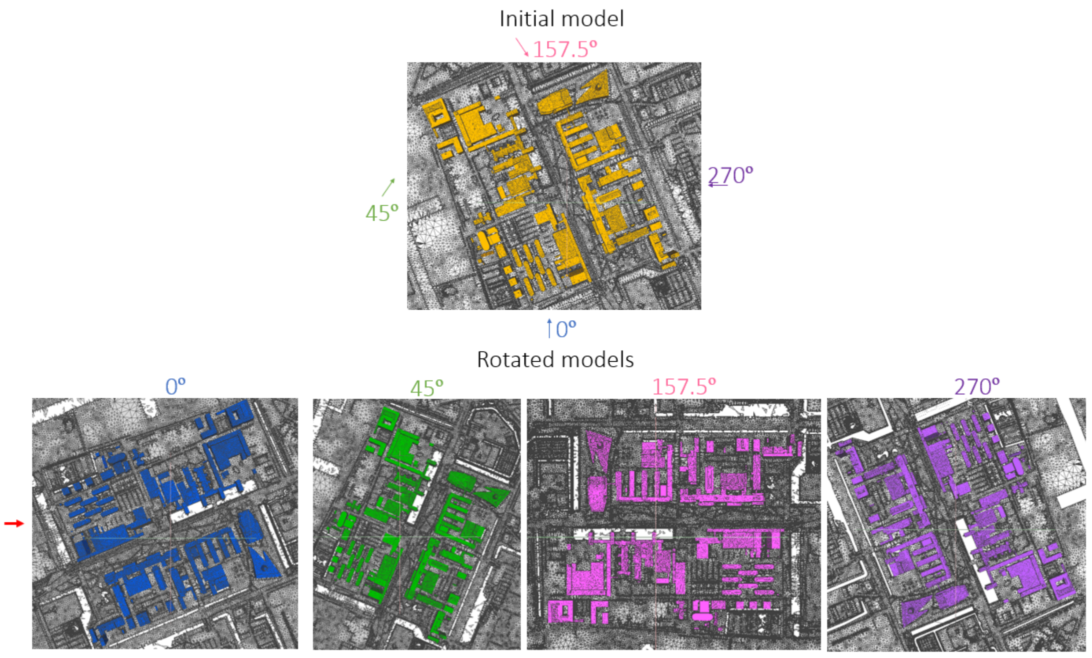


Figure 5.26.: Model rotation of the TU Delft campus 2. White surfaces in the rotated model are water surfaces that have not rotated with the rest of the model.

5.2.2. Computational domain

To test whether the method adheres to the required minimum dimensions for the domain, three models were selected with different h_{max}: Maastoren (h_{max} = 168.05m), Centre of Delft (h_{max} =72.01m), and Ouddorp (h_{max} = 9.40m). Tables 5.3, 5.4, and 5.5 present the results.

The domain dimensions computation was executed multiple times:

- Without blockage ratios,
- With blockage ratio BR defined by Franke and Baklanov (2007),
- With blockage ratios in the lateral horizontal BR_L and vertical BR_H directions defined by Blocken (2015).

The following can be observed:

- With or without blockage ratios, the minimum domain dimensions are maintained in each model: there should be a minimum of $20 \cdot h_{max}$, $10 \cdot h_{max}$, and $5 \cdot h_{max}$ in the x, y, and z directions, respectively.
- If the method applies BR, BR is respected in each case: BR should be lower than or equal to 3%.
- If the method uses BR_L and BR_H they are both respected in each case: both BR_L and BR_H should be lower than or equal to 17%.

The method works as expected. However, Figure 5.27 shows that the domains respecting BR or both BR_L and BR_H are very large in the y-direction. The question is whether this affect the simulation results, as the CFD guidelines suggest a larger distance between the urban model and the outflow boundary than the one between this model and the lateral boundaries (Section 2.2.1). Also, their total sizes appear to be much larger than the domain without BR and since one of the objectives is to create a mesh parameters definition method for average users, this might be an issue.

Table 5.3.: Suggested domain dimensions for Maastoren, depending on the chosen blockage ratio (BR) (h_{max} (x), (y), (z): number of h_{max} in the x, y, and z direction; BR: blockage ratio (Franke and Baklanov, 2007), BR_L: blockage ratio in the lateral horizontal direction (Blocken, 2015), and BR_H: blockage ratio in the lateral vertical direction (Blocken, 2015)).

	no BR	BR	BR_L and BR_H
3 h_{max} (x)	20.06	20.04	20.04
h_{max} (y)	10.07	19.77	31.24
h_{max} (z)	5.03	7.12	5.12
BR	0.07	0.03	0.03
BR_L	0.39	0.25	0.17
BR_H	0.17	0.12	0.17

Table 5.4.: Suggested domain dimensions for the Centre of Delft, depending on the chosen blockage ratio (BR) (h_{max} (x), (y), (z): number of h_{max} in the x, y, and z direction; BR: blockage ratio (Franke and Baklanov, 2007), BR_L: blockage ratio in the lateral horizontal direction (Blocken, 2015), and BR_H: blockage ratio in the lateral vertical direction (Blocken, 2015)).

	no BR	BR	BR_L and BR_H
h_{max} (x)	20.09	20.05	20.09
h_{max} (y)	10.04	26.79	35.83
h_{max} (z)	5.07	6.30	5.07
BR	0.07	0.03	0.03
BR_L	0.42	0.22	0.17
BR_H	0.17	0.14	0.17

Table 5.5.: Suggested domain dimensions for Ouddorp, depending on the chosen blockage ratio (BR) (h_{max} (x), (y), (z): number of h_{max} in the x, y, and z direction; BR: blockage ratio (Franke and Baklanov, 2007), BR_L: blockage ratio in the lateral horizontal direction (Blocken, 2015), and BR_H: blockage ratio in the lateral vertical direction (Blocken, 2015)).

	no BR	BR	BR_L and BR_H
h_{max} (x)	20.14	20.14	20.14
h_{max} (y)	10.29	248.30	317.22
h_{max} (z)	5.75	5.75	5.75
BR	0.13	0.03	0.03
BR_L	0.86	0.21	0.17
BR_H	0.16	0.16	0.17

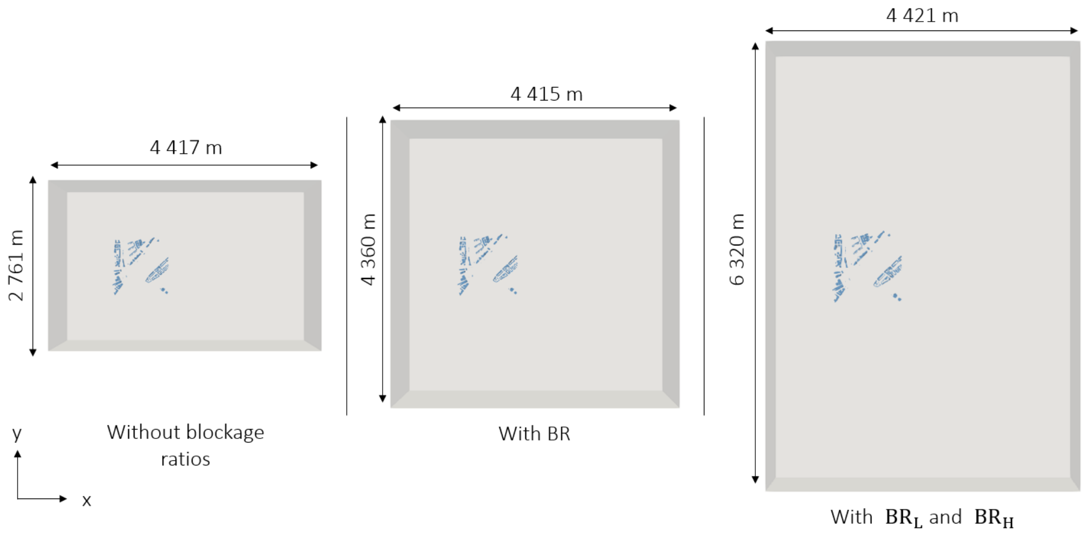


Figure 5.27.: Recommended domain dimensions depending on the chosen blockage ratio (BR) for Maastoren (BR: blockage ratio (Franke and Baklanov, 2007), BR_L: blockage ratio in the lateral horizontal direction (Blocken, 2015), and BR_H: blockage ratio in the lateral vertical direction (Blocken, 2015)).

To analyse the effect of the height of the tallest building h_{max} on the mesh parameters definition, mesh parameters for the same three models were computed with a limitation N_{max} of 30 million cells and roughness heights values z_0 of 0.5 and 2m. Table 5.7 demonstrates that the cell dimensions are most impacted by the roughness value. In fact, the same cell sizes were advised for the three models when z_0 was 2. However, the domain dimensions differ significantly as they depend on h_{max}, which was expected (Section 5.2.2). The model with the highest building, Maastoren, ends up having the largest domain. Table 5.6 shows that a lower roughness value results in greater cell dimensions in models with higher h_{max}, which can be explained by the maximum number of cells allowed. Large domains need larger cells to achieve the same density as smaller domains.

Table 5.6.: Influence of h_{max} on the cell and domain dimensions ($N_{max} = 30$ million cells, z_0 = 0.25m).

Dataset	Maastoren	Delft	Ouddorp
h_{max} (m)	168.05	72.01	9.40
hxw (m)	12.9x12.84	5.4x5.39	4.8x4.01
V_{domain} (m^3)	$1.24 \cdot 10^{10}$	$1.08 \cdot 10^{9}$	$5.38 \cdot 10^{7}$

Table 5.7.: Influence of h_{max} on the cell and domain dimensions (N_{max} = 30 million cells, z_0 = 2m).

Dataset	Maastoren	Delft	Ouddorp
h_{max} (m)	168.05	72.01	9.40
hxw (m)	32x26.68	32x26.68	32x26.68
V_{domain} (m^3)	$1.26 \cdot 10^{10}$	$1.12 \cdot 10^9$	$6.15 \cdot 10^6$

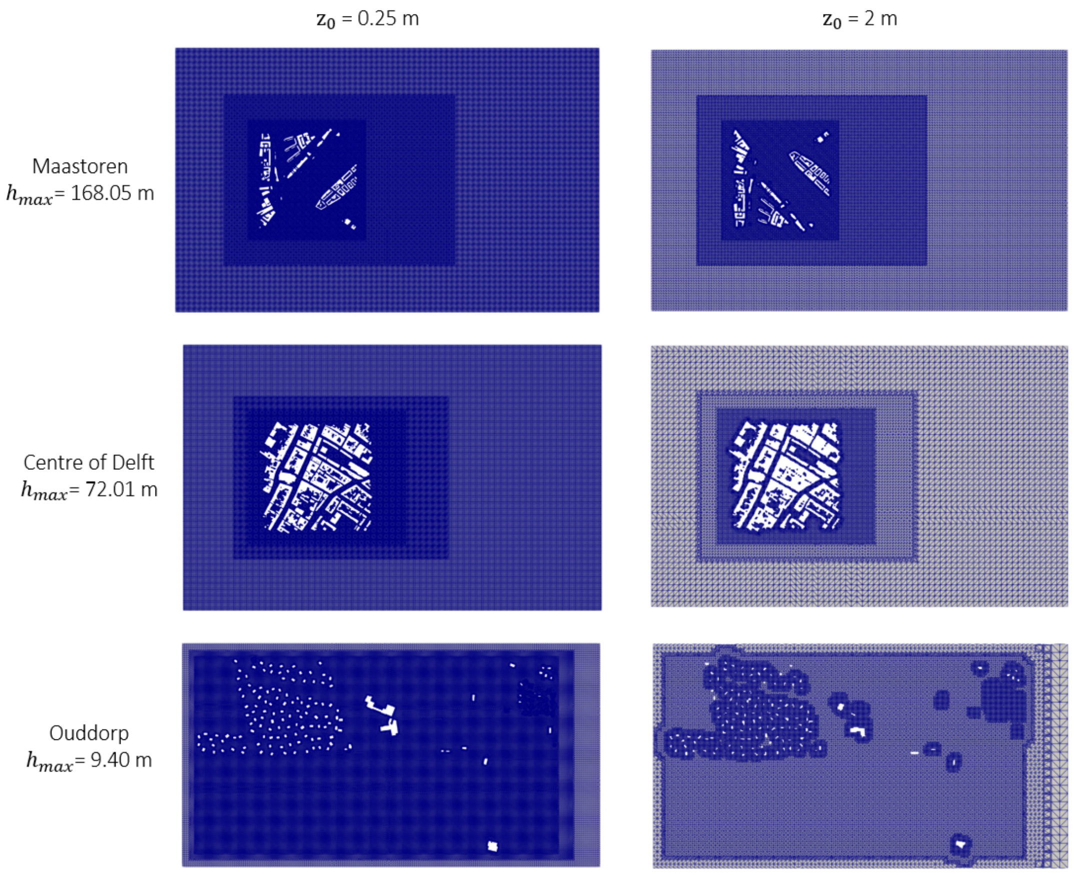


Figure 5.28.: Meshes for models with different h_{max}. Roughness height values z_0 of 0.25 and 2m are used.

The prototype gives the option to select the largest building dimension d_{max} instead of the height of the largest building h_{max} to define the domain dimensions. This would solve issues that occur using the Ouddorp model. When h_{max} is used, the model is very close to the domain boundaries (Figure 5.28), as the height of the buildings is very small. This suggestion of using d_{max} is mentioned by Liu et al. (2018).

5.2.3. Evaluation height

Table 5.8 shows that the target evaluation height h_{user} influences the cell dimensions. Greater evaluation heights seem to result in larger cell dimensions. Also, h_{user} is only respected in cells with the highest refinement level. Therefore, a refinement grid should be added to the ground. In addition, this table demonstrates that with lower h_{user} values, h_{user} might not be satisfied as the first evaluation node must be at least z_0 from the ground. In this test case, where z_0 is set to 0.5, it leads to a minimum cell height of 2.5 ($z_0 \cdot 2 \cdot 2.5$). Consequently, achieving h_{user} becomes impossible, even with the most refined cells. In short, the evaluation height seems to always be respected as long as the roughness height allows it. Thus, the method seems to work correctly for the evaluation level.

Table 5.8.: Influence of the target evaluation height h_{user} on the cell dimensions and final evaluation heights (N_{max} = 30 million, z_0 = 0.5m). h_{user} (0-3) indicates the final maximum h_{user} (cells with refinement level 0) and minimum h_{user} (cells with refinement level 3).

h_{user} (m)	1.5	3	5	10	20
hxw (m)	8x6.60	9.6x8.01	16x13.35	32x26.68	64x53.35
h_{user} (m) (0-3)	20-2.5	24-3	40-5	80-10	160-20

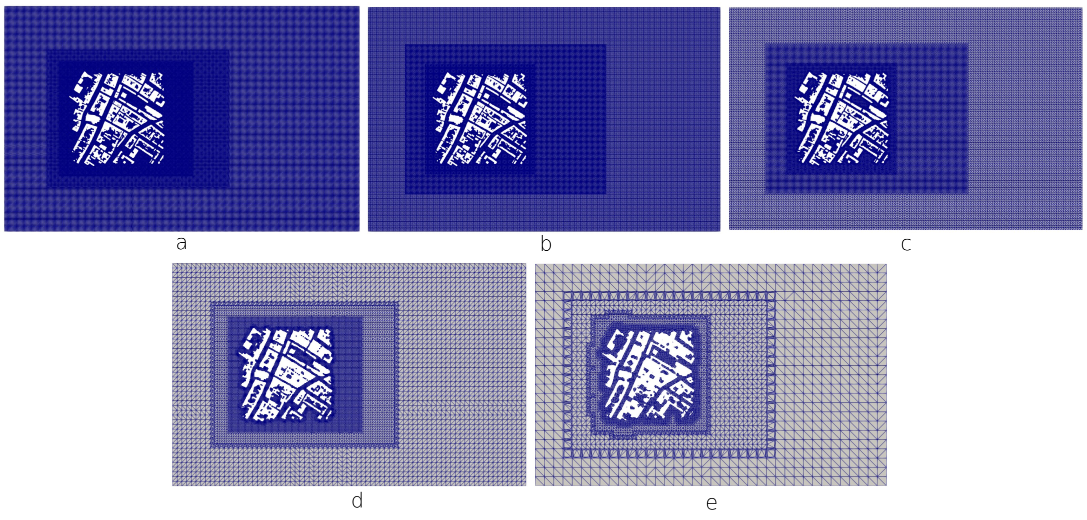


Figure 5.29.: Meshes for the Centre of Delft based on different target evaluation heights h_{user}: (a) 1.5m, (b) 3m, (c) 5m, (d) 10m, and (e) 20m.

5.2.4. Roughness height

The evaluation nodes, which are placed at the centre of the cells, must be located at a minimum distance equal to the roughness height z_0 from the ground. This requirement previously mentioned is the most important one within the mesh parameters definition, and that is why it should always be respected. Appendix F demonstrates that this condition was met in each test case.

To analyse the impact of the roughness height, the mesh parameters definition was performed on the Centre of Delft with different values of z_0. The results are presented in Table 5.9 and Figure 5.30. First, it appears that cell dimensions increase with higher z_0 values, which is as expected. Second, while the height of the lowest evaluation nodes exceeds z_0, the cell dimensions remain the same with z_0 values between 0.25 and 0.03. This is probably due to the method trying to maintain the target evaluation height of 1.5m at the 3^{rd} cell from the ground.

Table 5.9.: Resulting cell dimensions hxw, distances between the bottom boundary and first evaluation nodes h_{node}, and number of cells approximation $N_{approx.}$ for the Centre of Delft (N_{max} = 30 million). The distance between the bottom boundary and first evaluation nodes from this boundary is denoted h_{node}.

z_0 (m)	0.03	0.10	0.25	0.5	1	2
hxw (m)	5.4x5.39	5.4x5.39	5.4x5.39	8x6.68	16x13.35	32x26.68
h_{node} (m)	0.34	0.34	0.34	0.5	1	2

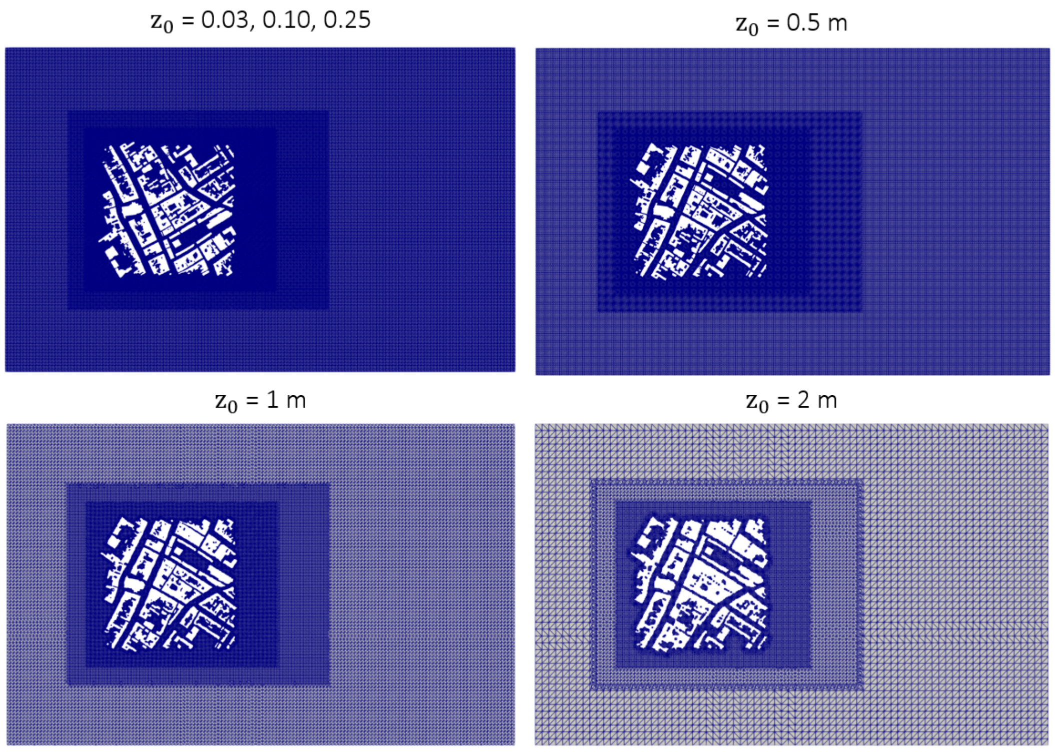


Figure 5.30.: Suggested meshes for the Centre of Delft with different roughness height values z_0.

5.2.5. Maximum number of cells

The tests in Appendix F show that the number of cells limit is not exceeded with respect to the approximation of the number of cells. However, this is not always true for the number of cells in the final mesh generated by *snappyHexMesh*. On average, based on these tests, the approximation deviates from the final cell count by -12.12%. It seems to be difficult to determine why there is more difference in some cases than in others as OpenFOAM applies some optimisation tasks (e.g. local refinements) hard to understand during the meshing process. Users would be informed by this average accuracy.

The 3D models representing Ouddorp and the Centre of Delft were used to test the mesh parameters definition method with different number of cells limits N_{max}. Tables 5.10 and 5.11 present the results. With a lower limit, the cell dimensions increase, which can be explained by the fact that dividing the domain by a lower number results in larger cells. These tables show that after a certain cell limit, the cell dimensions remain constant. This is probably due to the roughness height and/or target evaluation height requirements.

Table 5.10.: Recommended cell dimensions for Ouddorp with different limits for the number of cells N_{max} (z_0 = 0.5m, h_{user} = 1.5m).

N_{max}	1000	10^6	$10 \cdot 10^6$	$30 \cdot 10^6$	$50 \cdot 10^6$	$80 \cdot 10^6$	$100 \cdot 10^6$
hxw (m)	285.7x285.69	12.1x11.3	8x6.68	8x6.68	8x6.68	8x6.68	8x6.68

Table 5.11.: Recommended cell dimensions for the Centre of Delft with different limits for the number of cells N_{max} ($z_0 = 0.5$m, $h_{user} = 1.5$m).

N_{max}	1000	10^6	$10 \cdot 10^6$	$30 \cdot 10^6$	$50 \cdot 10^6$	$80 \cdot 10^6$	$100 \cdot 10^6$
hxw (m)	432.3x432.26	19x19	8x7.93	8x6.68	8x6.68	8x6.68	8x6.68

The mesh parameters definition primarily focuses on models without terrain; however, a method was also developed for models with terrain features. For this reason, the number of cells approximation was also estimated for the TU Delft campus 2 file, which include terrain surfaces, and three different cell dimensions. Table 5.12 shows the difference between the approximated and final number of cells. As can be observed, the approximation seems to be more accurate with small cell dimensions. However, the method needs further validations before assessing its accuracy. Therefore, it should be tested with different 3D urban models.

Table 5.12.: Number of cells approximation for TU Delft campus 2.

hxw (m)	8x6.68	11.9x11.84	26.7x26.67
$N_{approx.}$	68 439 813	17 404 501	2 406 529
N_{final}	69 908 126	20 072 872	2 975 629
Difference (%)	-2	-13	-19

5.2.6. At least 10 cells per cube root of the building volume

In the final mesh, there must be at least 10 cells per cube root building volume. To test whether the algorithm identifies correctly the buildings not meeting this requirement, three models were tested.

First, a model containing 21 cubes with identical dimensions (150x100x125) was tested. The results are presented in Table 5.13. The cell dimensions shown in this table were defined based on their cube root volume of 123.31m. To fit 10 cells within this value, the cell height and width should be approximately 12.33m. As the method evaluates this requirement with the most refined cells (level 3 in this case), the maximum cell dimensions without refinement should be (12.33x8)x(12.33x8). Observe that in the two first cases, where the cell heights and widths are below $12.33 \cdot 8$, the cubes were considered valid. However, the other cases result in invalid cubes, as the cell dimensions exceed this value. Also, the last two cases demonstrate that both the width and height can have a maximum of $12.33 \cdot 8$; otherwise, the cubes are considered invalid.

Table 5.13.: Number of valid buildings identified in the model with 21 identical cubes. Cube root of one cube ($\sqrt[3]{Cube}$) equals 123.31m.

w_{cell}	h_{cell}	$10 \cdot w_{cell}$	$10 \cdot h_{cell}$	Valid
10x8	10x8	100	100	100%
12.30x8	12.30x8	123	123	100%
12.34x8	12.34x8	123.4	123.4	0%
12.30x8	12.34x8	123	123.4	0%
12.34x8	12.3x8	123.4	123	0%

Second, the models of Ouddorp and TU Delft campus were tested, which contains mostly small and large buildings, respectively. For each test case, the mesh parameters definition was applied with three different roughness heights z_0 as they have the greatest influence on the cell dimensions (Section 5.2.4). Figure 5.31 shows the resulting invalid buildings. In each case, an increase of z_0 results in the selection of more invalid buildings. However, it can be observed that the number of buildings identified in Ouddorp remains relatively consistent compared to the model representing the TU Delft campus. This contrast can be explained by the number of small buildings in Ouddorp and the amount of large buildings within the TU Delft campus model.

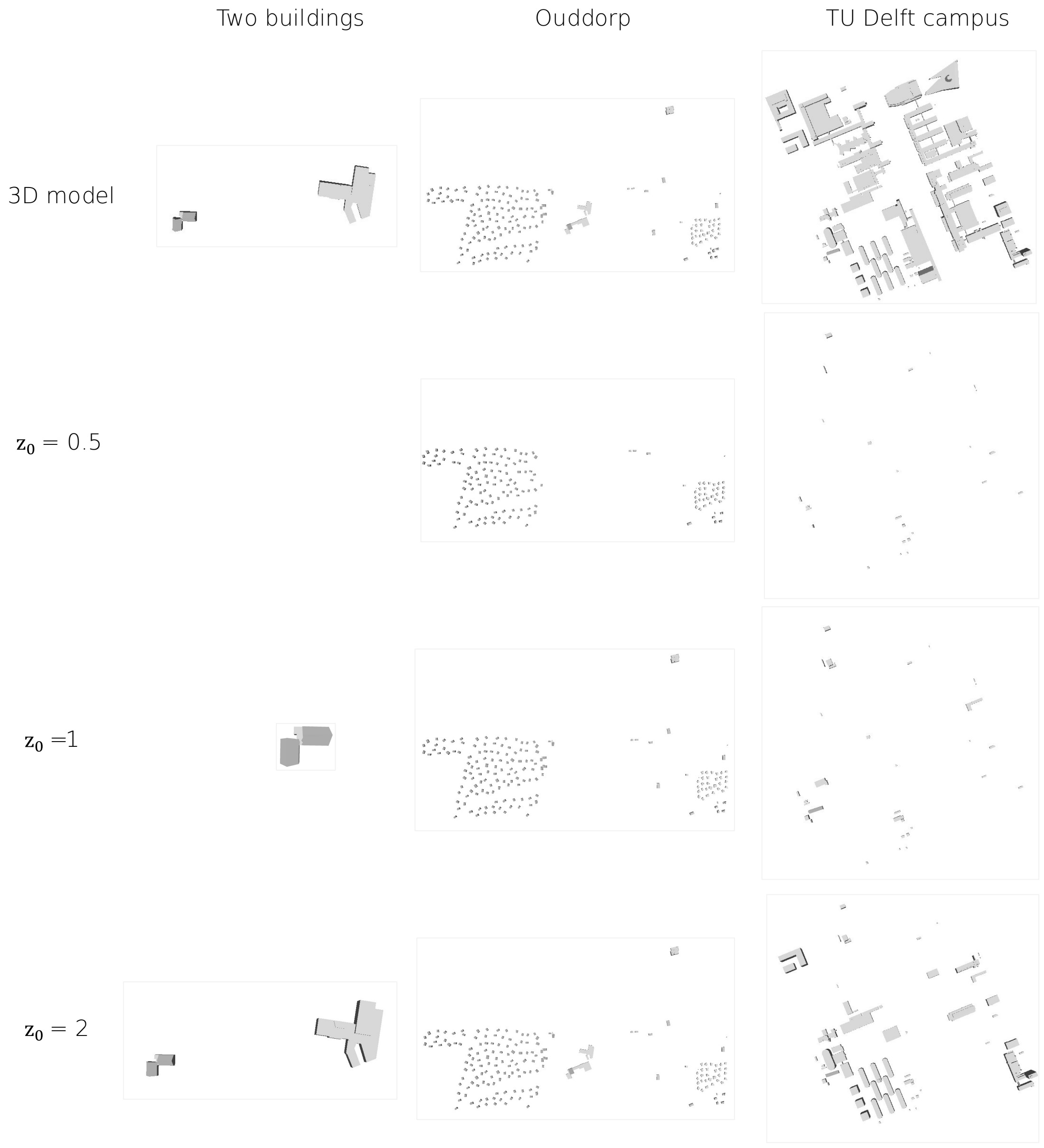


Figure 5.31.: Number of buildings with less than 10 cells per cube root building volume ($\sqrt[3]{buildingvolume}$) within real-world datasets.

From the tests above, the method seems to correctly identify buildings with insufficient cells per cube root building volume. In addition, the test cases presented in Appendix F show that the cell ratio, which was set to a maximum of 1.20, is always respected.

5.2.7. Building distribution

CFD guidelines recommend to maintain at least 10 cells per building separation. To verify whether the method detects buildings that are not separated with sufficient cells, three models were tested.

First, the method was performed multiple times with the same model as described in Section 5.2.6, containing 21 cubes separated by distances of 100 (d_x) and 75 (d_y) in the x and y directions, respectively. Figure 5.32 shows the results. Note that the focus does not lie on diagonal distances which are equal to 125. Given that a geometry is surrounded by a minimum of 4 rows with the most refined cells (level 3 in this case), there must be a minimum of 8 cells with refinement level 3 and 2 cells with refinement level 2 to ensure a number of 10 cells between two buildings. Based on this information, separation thresholds $d_{separation}$, which exclude separations below this specified value, were defined. In addition, to respect the requirement mentioned above for d_x and d_y, the cell width at refinement level 3 should be 6.25 and 8.33, respectively. This leads to maximum cell widths without refinement w_{cell} of $6.25 \cdot 8$ and $8.33 \cdot 8$.These values were used to define w_{cell} in Figure 5.32. The following can be observed:

- (a): Since $d_{separation}$ is higher than d_x and d_y, these separations were not taken into account and all cubes/buildings were considered valid.
- (b) and (c): w_{cell} is higher than the maximum w_{cell} for both directions. In case (b), cubes were labelled invalid only due to d_x (d_y was not taken into account). In case (c), cubes were considered invalid due to both separations.
- (d): cubes were considered invalid due to w_{cell}, which was higher than $6.25 \cdot 8$ but lower than $8.33 \cdot 8$.
- (e) and (f): all cubes are valid, w_{cell} is lower than both $6.25 \cdot 8$ and $8.33 \cdot 8$.

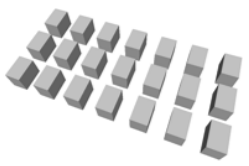
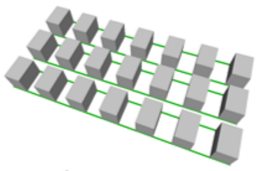
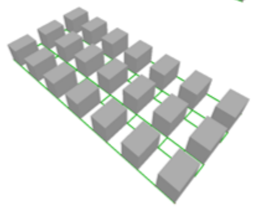
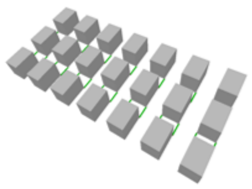
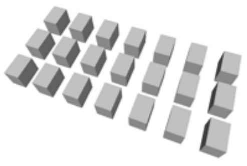
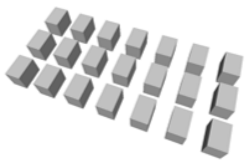
	w_{cell}	$d_{separation}$	min	score	Invalid building separations
a	9x8	200	108	100%	
b	9x8	100	108	0%	
c	9x8	70	108	0%	
d	6.5x8	70	78	0%	
e	6.25x8	70	75	100%	
f	6.x8	70	72	100%	

Figure 5.32.: Number of invalid building separations, indicated in green, in the model with 21 identical cubes (w_{cell}: cell width, $d_{separation}$: separation threshold, *min*: minimum distance needed, which is $(w_{cell}/8)$*8+$(w_{cell}/4)$*2, and score: percentage of valid buildings). Cubes are separated with distances of 75, 100, and 125m.

Figure 5.33 illustrates the results obtained from two other models: one representing the Begijnhofbuurt in Amsterdam (high building density), and one representing Ouddorp (low building density). Several observations can be made. First, as the roughness height decreases, fewer buildings are classified as invalid. Second, the number of invalid distances decreases as the value of $d_{separation}$ increases. Finally, the number of invalid buildings seems to be higher in the Begijnhofbuurt model compared to the Ouddorp model, which can be explained by the higher building density of Begijnhofbuurt.

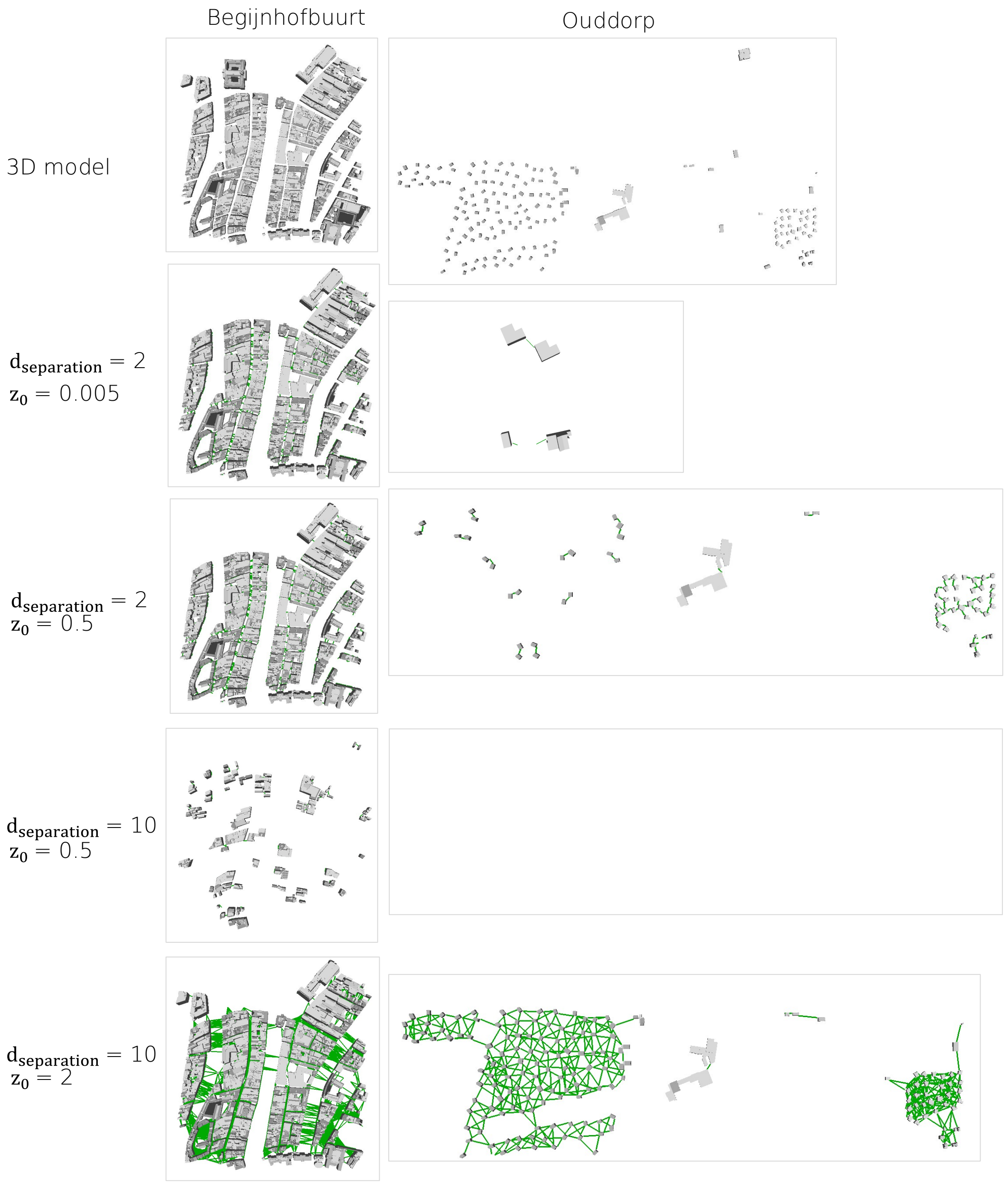


Figure 5.33.: Invalid buildings separations, indicated in green, within the Begijnhofbuurt (Amsterdam) and Ouddorp files.

The method appears to correctly identify buildings with separations with insufficient cells. In addition, the test cases presented in Appendix F show that the cell ratio, which was set to a maximum of 1.20, is always respected. However, the number of buildings following this requirement appears to be relatively low. In fact, it varies between 0.58% and 21.18%. Yet, this can be expected, as it is nearly impossible to respect all CFD guidelines.

5.2.8. Region of Interest

The Region of Interest (RoI) was tested using the model representing Ouddorp. First, the RoI was generated without a target building. Figure 5.34 (a) demonstrates that the algorithm selects the centre of the 3D model as the centre of the RoI. Subsequently, the RoI was generated with a target building. Figure 5.34 (b) shows that the centre of the RoI is on the target building and that the radius of the RoI is significantly smaller. This is due to the algorithm that considers the highest dimension of the target building as the radius of RoI when there is a target building; otherwise, it takes the highest dimension of the buildings within the 3D model. As this target building is much smaller than the largest building within the 3D model, the resulting RoI is relatively small.

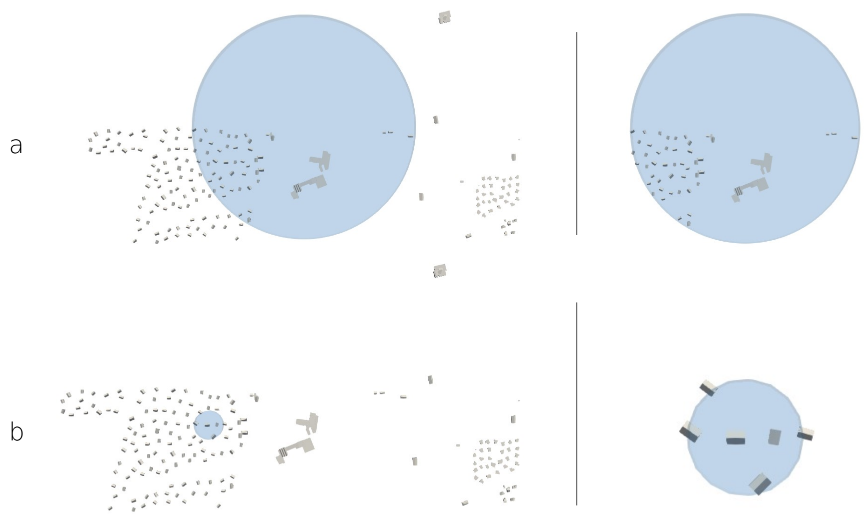


Figure 5.34.: Region of Interest definition with Ouddorp (a) without target building (b) with target building.

5.2.9. Ground refinement

Tables 5.14 and 5.15 present the results obtained when a ground refinement is applied to the Centre of Delft and Ouddorp. When the ground refinement is applied without cell limitation, the cell dimensions remain the same and the number of cells in the mesh increases, which is expected. When a cell limitation is set, the cell dimensions may increase to respect this maximum, as demonstrated in Table 5.14. On average, the cell approximation deviates by -22.2% from the final cell count (Appendix F), which is worse than the approximation without ground refinement. Similar to the latter approximation, it is difficult to make an accurate approximation of the final cell number as OpenFOAM applies some optimisation tasks hard to understand.

Table 5.14.: Comparison between approximated number of cells $N_{approx.}$ and final number of cells N_{final} with or without ground refinement (GR) for the Centre of Delft (N_{max} = 30 million cells, z_0 = 0.25m).

	No GR	GR, no N_{max}	GR, with N_{max}
hxw (m)	5.4x5.39	5.4x5.39	6.2x6.16
$N_{approx.}$	29 515 237	40 503 050	29 736 376
N_{final}	44 194 102	65 682 877	40 976 798

Table 5.15.: Comparison between approximated number of cells $N_{approx.}$ and final number of cells N_{final} with or without ground refinement (GR) for Ouddorp (N_{max} = 30 million cells, z_0 = 0.25m).

	No GR	GR, no N_{max}	GR, with N_{max}
hxw (m)	4.8x4.01	4.8x4.01	4.8x4.01
$N_{approx.}$	20 575 779	22 467 191	22 467 191
N_{final}	18 841 020	29 792 968	29 792 968

8

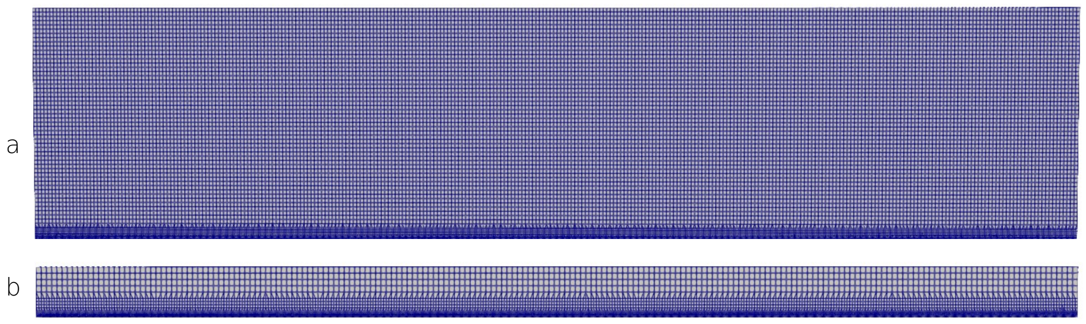


Figure 5.35.: Profile view of meshes with ground refinement (z_0 = 0.5m): (a) Centre of Delft (b) Ouddorp.

5.2.10. Number of refinement boxes

The test cases focus primarily on two refinement boxes; however, the prototype gives the option to users to select three refinement boxes. To verify whether this option works properly, some meshes with three refinement boxes were generated for the Centre of Delft. Figure 5.36 shows one of these meshes, and demonstrates that there are indeed three meshes defined. Table 5.16 compares two meshes with the same input parameters, except for the number of refinement boxes. Observe that the cell dimensions are greater with three refinement boxes, which is expected as there are refinement levels. The final numbers of cells are approximately the same, however, the number of cell approximation for three refinement boxes seems to be less accurate. This should be further explored.

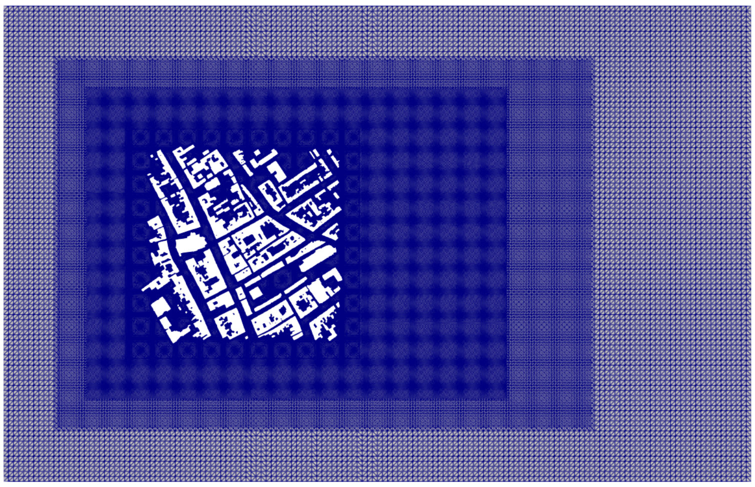


Figure 5.36.: Mesh generated for the Centre of Delft with three refinement boxes.

Table 5.16.: Comparison between two meshes suggested for the Centre of Delft. The same input parameters are used, except for the number of refinement boxes.

	Two refinement boxes	Three refinement boxes
hxw (m)	8x6.68	16.1x13.43
$N_{approx.}$	13 629 727	27 509 576
N_{final}	17 601 037	19 167 071
Difference (%)	-22	-30

5.3. Comparison between CFD simulations with mesh parameters of an OpenFOAM tutorial and the prototype

To demonstrate the difference between CFD simulations with and without CFD guidelines for urban areas, two CFD simulations were performed. The first one uses mesh parameters from an OpenFOAM tutorial named *windAroundBuildings*, which simulates wind flows around a set of buildings presented in Section 4.2 (”Buildings”); the second one includes mesh parameters suggested by the prototype.

A modified version of the *windAroundBuildings* tutorial was used for both CFD simulations. First, Atmospheric Boundary Layer (ABL) conditions were added by including the *ABLconditions* directory from another tutorial named *turbineSiting*. Additionally, the following parameters were configured:

- Roughness length z_0: 0.5 m
- Velocity value U_{ref}: 5 m/s (in the x-direction)
- Velocity height z_{ref}: 10 m

- Turbulent kinetic energy κ[1]: 1.5
- Passive scalar ϵ[2]: 0.03

The next step was to run the simulations, using this modified version of the tutorial including the input file "Buildings", and the two sets of mesh parameters (i.e. different sets of *blockMeshDict* and *snappyHexMeshDict* files). To perform them, the *simpleFoam* solver was used, which solves incompressible and turbulent flows in steady-state situations by using the SIMPLE algorithm (OpenFOAM, nd). SIMPLE is a widely used approach to solve equations of fluid mechanics (Khawaja and Moatamedi, 2018).

Figure 5.37 illustrates the simulation results in 3D. The domain dimensions applied by the tutorial (a) are much smaller than the ones recommended by the prototype (b) and do not follow the CFD guidelines. The latter seems to lead to simulation errors as Figures 5.38 and 5.39 show some differences in the velocity values around the buildings facing the incoming flow (on the left). The flows from the tutorial (Figure 5.38) have higher velocity values (8-10 m/s) compared to the flows from the prototype (approx. 6 m/s) (Figure 5.39). These higher values are probably due to insufficient upstream domain length (distance between the inlet boundary and building model).

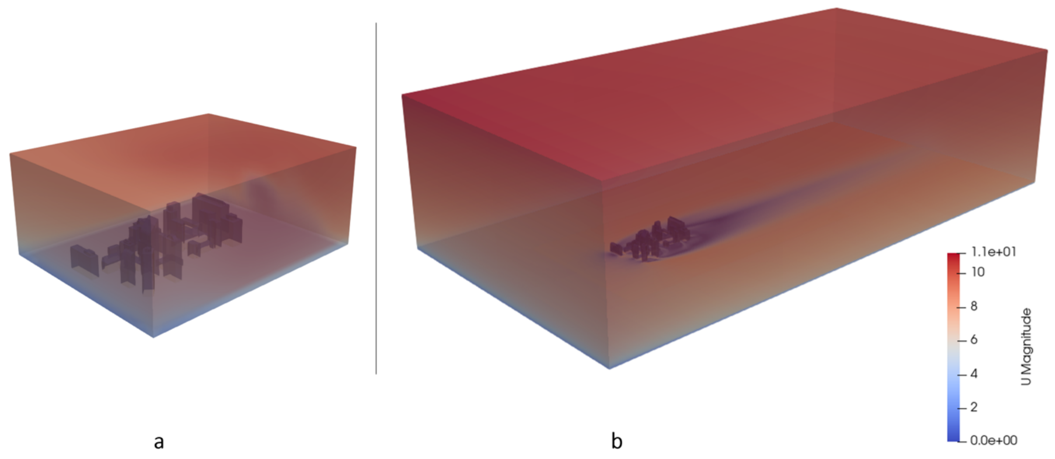


Figure 5.37.: 3D view CFD simulations with mesh parameters from (a) Tutorial (b) Prototype.

[1]κ: energy possessed by turbulent flows resulting from motion (Britannica, 2023).

[2]ϵ: quantity convected by fluids without influencing their dynamic behaviours, such as temperature or salinity (Kerr, 1981).

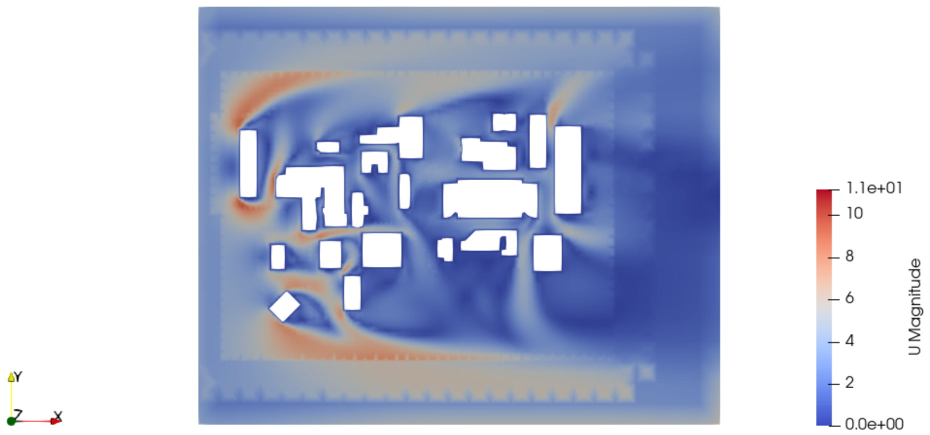


Figure 5.38.: Slide at 5m of CFD simulation with mesh parameters from tutorial.

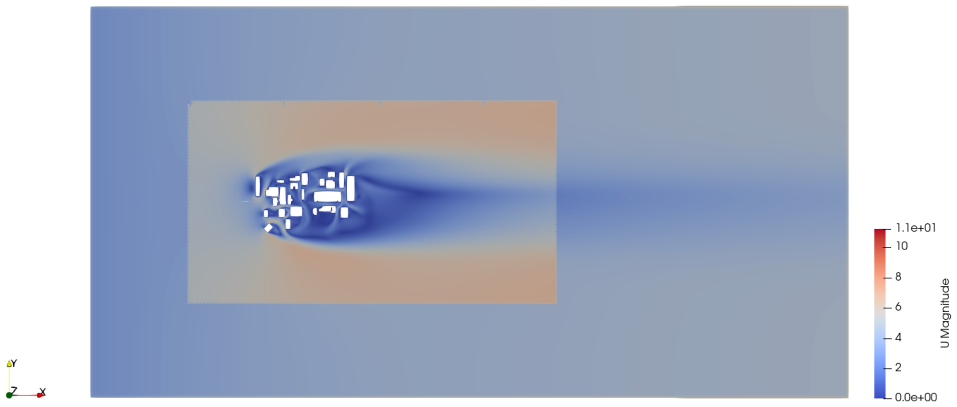


Figure 5.39.: Slide at 5m of CFD simulation with mesh parameters from prototype.

Figures 5.40 and 5.41 illustrates the resulting meshes from the tutorial and prototype, respectively. It shows that the tutorial used larger cells, with its background mesh containing cells of 14x14, while the prototype suggests cells of 8.01x8. Compared to the tutorial, the prototype contains 282 times more cells in the background mesh and 23 times more cells than the final mesh. This difference lead to more accuracy in the mesh of the prototype, as the tutorial applies a smaller domain and larger cells. Furthermore, the tutorial uses one refinement box instead of two. These boxes allow to find a balance between accurate results and an adequate computational performance. Smaller cells often lead to more accurate results but can result in computationally expensive simulations. They enable smooth transitions between flow values inside the region of interest (around buildings) and remaining area of the domain. Applying more refinement boxes improves the smoothness of these transitions.

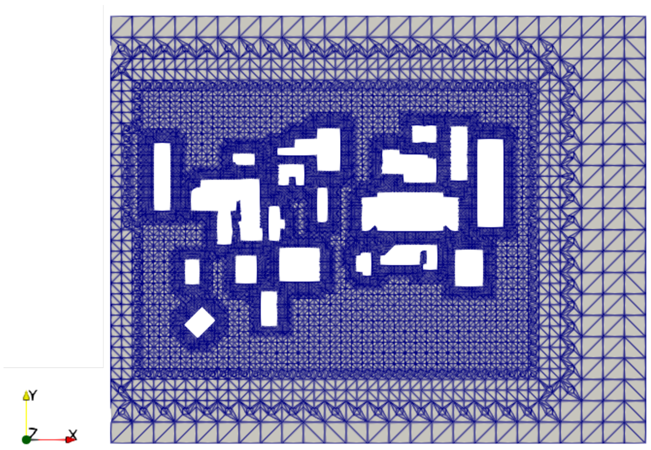


Figure 5.40.: Mesh tutorial (slide at 5m). Background mesh (*blockMeshDict*) and final mesh (*snappyHexMeshDict*) contain 5000 and 185 749 cells, respectively.

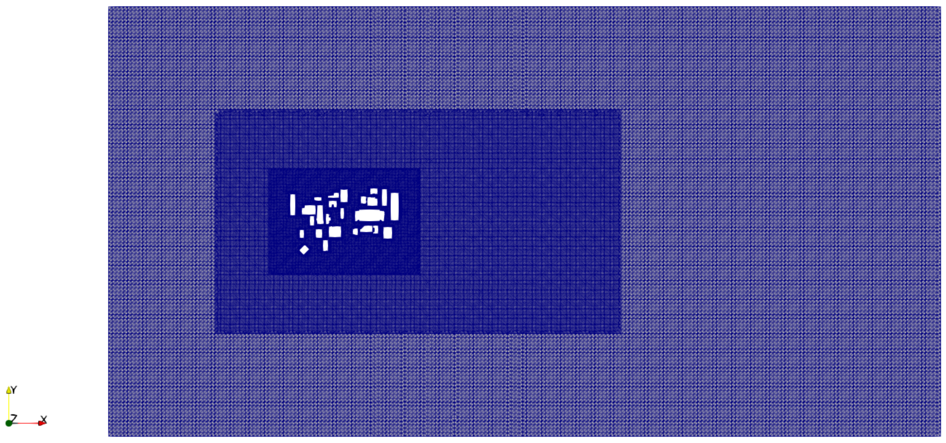


Figure 5.41.: Mesh prototype (slide at 5m). Background mesh (*blockMeshDict*) and final mesh (*snappyHexMeshDict*) contain 1 410 579 and 4 328 987, respectively.

Figure 5.42 presents the velocity profiles of both simulations at the front of the urban model (Figure 5.43), which represents the magnitude of the velocity in relation with height. The velocity magnitude is 0 m/s at the ground and increases with the height. This figure demonstrates that smoother velocity profile of the prototype compared to that of the tutorial. This is due to the larger cells and lower number of refinement boxes applied in the tutorial, which lead to larger differences between cell/flow values. Additionally, it can be observed that the velocity profile of the tutorial stops at a lower height due to its smaller domain.

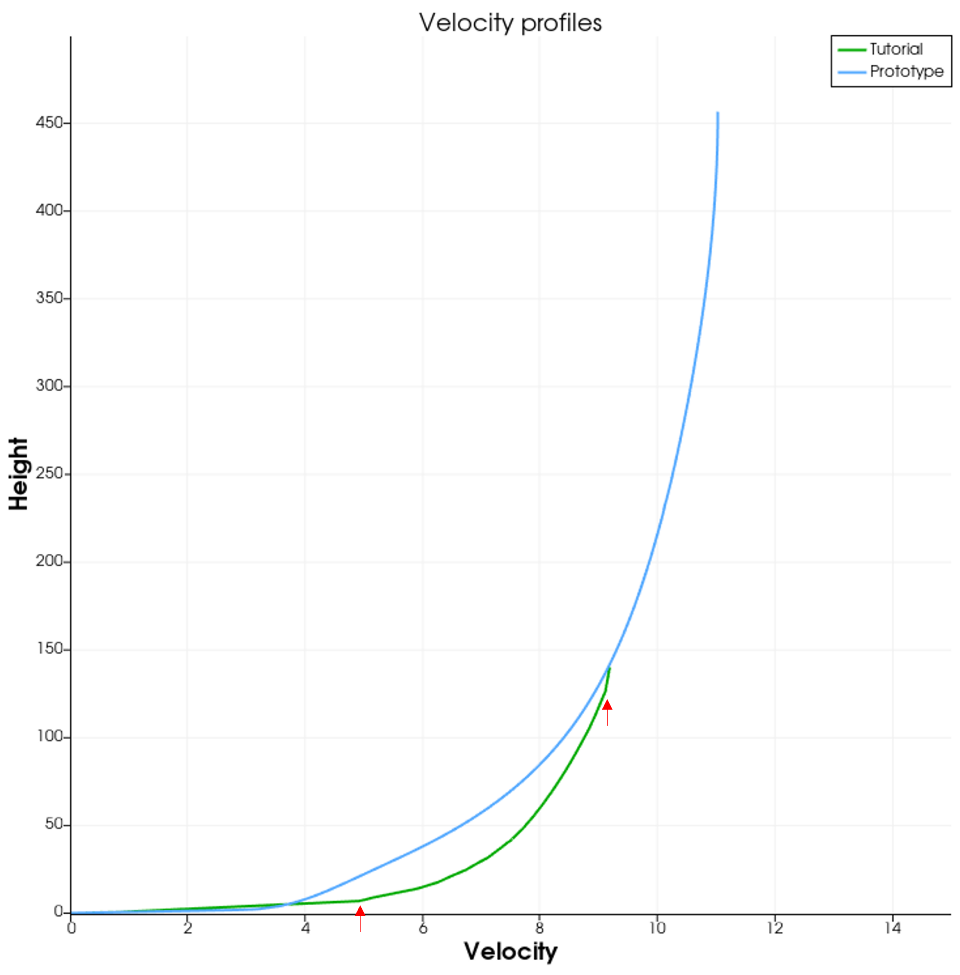


Figure 5.42.: Velocity profiles. Rough changes are indicated with red crosses.

Finally, the residuals were plotted to observe the convergence of velocity U, turbulent kinetic energy κ, and passive scalar ϵ values (Figures 5.44 and 5.45). The same behaviour can be observed for both simulations: the values decreases and converges to a certain value. However, the tutorial converges already around 500 iterations and the prototype a bit before 5000 iterations. The residual values are higher for the tutorial than the one of the prototype, which indicate a higher accuracy of the prototype.

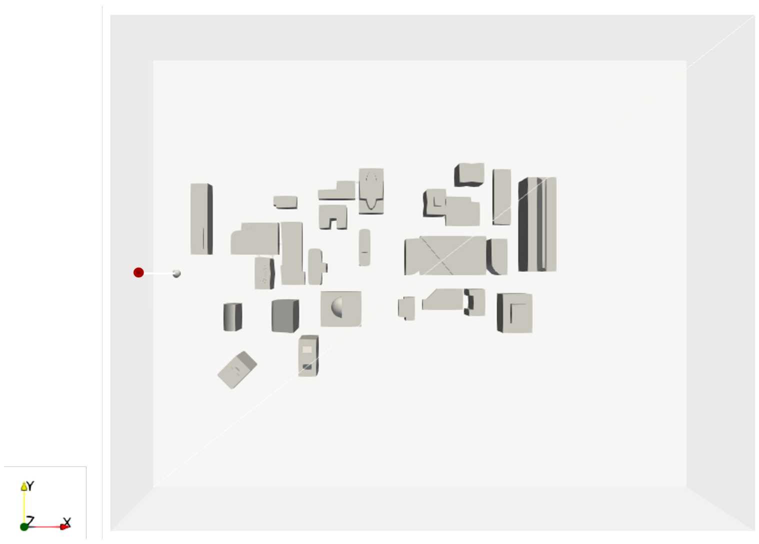


Figure 5.43.: Location at which the velocity was evaluated indicated in red.

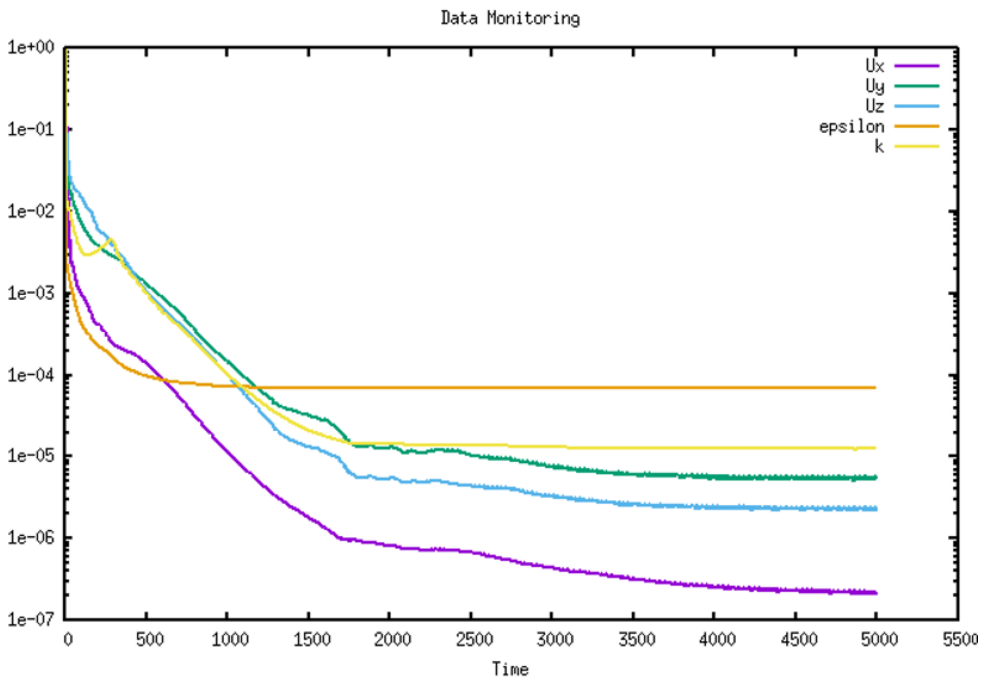


Figure 5.44.: Residuals of the prototype.

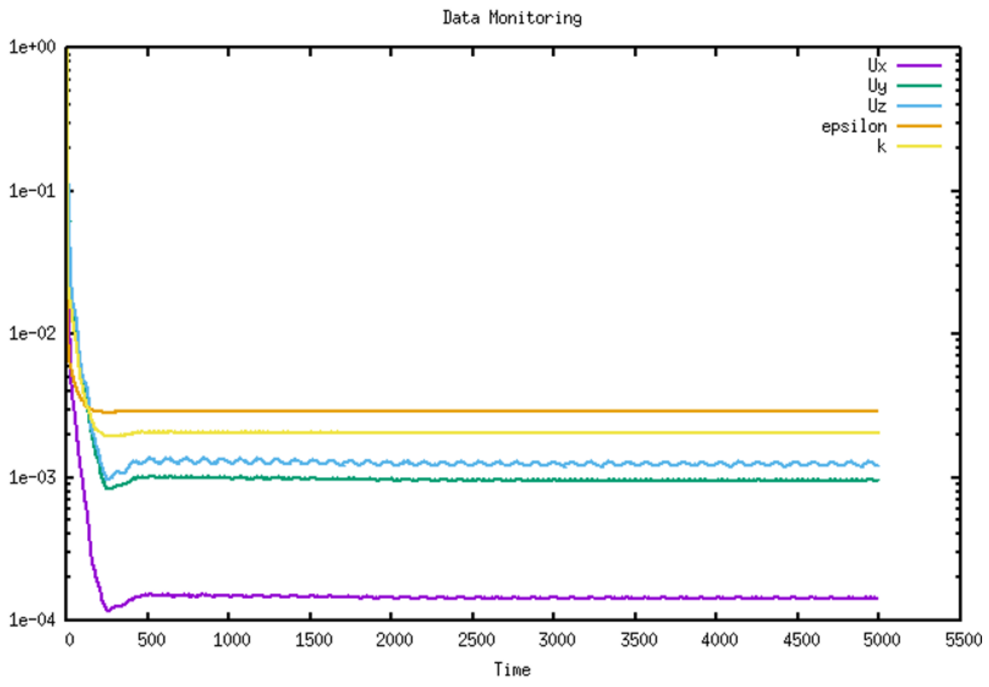


Figure 5.45.: Residuals of the tutorial.

To conclude, the results described above suggest that the mesh parameters of the prototype lead to more accurate results. However, its simulation time was much larger (3 hours and 10 minutes) than the one of the prototype (7 minutes).

6. Conclusions, discussion and recommendations

6.1. Conclusions

The aim of this thesis was to develop a method that validates geometries and defines mesh parameters to simplify the use of CFD simulations in urban areas. To reach this objective, a prototype was created, in the form of a web application, that executes this method. The following four research questions were explored during the thesis.

Which geometric validations should be performed for CFD simulations in urban areas? And how to implement them?

These questions were answered by literature review and by translating the identified validations into code. The chosen validations are described below.

Standards were developed to define basic primitives (ISO19107) (ISO, 2019) and provide guidelines for their digitization (OGC, 2016, 2011), with the aim of enhancing the interoperability and exchange of geographical data. As the quality of geometries plays an important role in CFD simulations, verifying them based on the ISO19107 standard could be a "useful starting point" (Wagner et al., 2015). Therefore, the validation tool named val3dity developed by Ledoux (2018) was implemented in the prototype. This was done by including the val3dity library in the C++ script of the prototype.

However, one important geometric validation is missing in this validation tool: the topological relationship between buildings and terrain features. Consequently, floating buildings over terrains are allowed, leading to incorrect simulation results (in reality, buildings are always connected to the ground). To address this, an additional validation was added that identifies buildings with ground surfaces higher than terrain surfaces (i.e. floating buildings), and buildings separated with a user-defined distance from these surfaces. In the absence of terrain features, the ground level is used instead, which can be user-defined or the lowest z-value.

Another validation involves identifying overlapping buildings, as they can affect the mesh quality. To do so, the intersections between buildings are computed. When their volume is non-zero, buildings are considered overlapping. However, this method cannot always be applied when buildings are incorrectly defined (not bounding a volume or self-intersecting). An alternative method was developed that calculates the area of the intersections between ground surfaces in 2D. If this area is non-zero, the buildings are identified as overlapping.

Finally, small geometric details might also affect the meshing process, including sharp angles, short edges, and sliver triangles. To help users simplify their geometries, algorithms were implemented that identify these shapes. These algorithms iterate over (triangular) faces, and evaluate the angles between them, their edge lengths, and sliver parameters. If these values are lower than a user-defined threshold, they are identified as one of these small geometries.

Which method(s) could be used to compute CFD mesh parameters and generate high quality meshes for urban CFD simulations in OpenFOAM?

This question was first investigated by literature review to select guidelines ensuring high quality meshes for CFD simulation in urban areas. Then, a method was developed that combines the chosen guidelines and translated into code.

Guidelines were developed to define accurate pre-run setups for CFD simulations in urban areas. Considering the scope of the thesis, only CFD guidelines related to the preparation of geometries and mesh parameters definition were discussed. Since not all guidelines can be satisfied at the same time, a selection must be made based on their relevance and implementation possibilities (i.e. potential to be formalized and translated to axioms and code, and possibilities within OpenFOAM). The selected guidelines cover the computational domain, computational grid (mesh), and Region of Interest (RoI).

Based on the chosen guidelines, a workflow was created that returns the following data to users:

- The 3D model aligned with the incoming flow simulated in OpenFOAM,
- Configuration files containing mesh parameters for CFD simulation in urban areas (*blockMeshDict* and *snappyHexMeshDict*),
- Advice and information on the input model aiming to help users find a balance between simulation accuracy and performance (e.g. number of buildings satisfying one of the guidelines, buildings needed to run a realistic simulation).

Figure 3.7, presented in Section 3.3.1, illustrates this workflow. More information on the methodology used can be found in Section 3.3.

How to report errors, warnings, and results to users?

This question was approached by first dividing the results from geometric validations and mesh parameters definition into errors, warnings, and advises (Table 6.1). This classification was based on their type of information and influence on CFD simulations. Second, a prototype of a web application was created presenting the results to users with their corresponding class. Some results are directly visible on the user interface; others are returned as OBJ files, TXT files, and/or configuration files for OpenFOAM and can be downloaded.

Table 6.1.: Classification of the results

Results	Class
Separate building and terrain validation (val3dity)	Indicated as error by the prototype, however, val3dity has its own classification.
Topological relationships between buildings and terrain	Error: floating building
	Warning: buildings too far below terrain surfaces or ground level
Sharp angles	Warning
Short edges	Warning
Sliver triangles	Warning
Overlapping buildings	Warning
Aligned 3D city model with incoming flow	Advice
Region of Interest (RoI)	Advice
Background mesh parameters (*blockMeshDict*)	Advice
Refined mesh parameters (*snappyHexMeshDict*)	Advice
Buildings satisfying required minimum of cells	Warning

How to validate the quality of the methodology?

Different methods were used to validate the quality of the methodology:

- **Simple shapes:** Algorithms performing geometric validations and computing CFD mesh parameters were tested using simple shapes. Some examples are cubes, cones and single buildings with or without terrain. This validation method allows to clearly identify and remove errors in these algorithms. When no more errors can be detected, one can conclude that their basis is solid.

- **Real-world datasets:** Some geometries are not directly considered during the development of algorithms, for example, faces with collinear vertices. For this reason, algorithms were also tested with real-world datasets, which can contain unexpected geometries. In this way, the algorithms can be improved to make them more robust and adaptable to different scenarios.

- **Mesh generations in OpenFOAM**: Several meshes were generated in OpenFOAM with different input files and parameters (e.g. roughness values, evaluation heights). The data from these meshes were compared to values computed by the prototype. These evaluations involved visual observations (with the 3D viewer ParaView) and numerical assessments (e.g. using the checkMesh command from OpenFOAM). This approach helped to verify the accuracy of the used methodology for mesh parameters definition.

- **Comparison between OpenFOAM tutorial and prototype:**: Two CFD simulations were performed: one with mesh parameters from an OpenFOAM tutorial and one with parameters suggested by the prototype (Section 5.3). The same input model and other simulation pre-run setups were used. As the tutorial does not follow CFD guidelines for the meshing process, their influence on the simulation results could be analysed. Thus, this approach provides insights in the contribution of the prototype.

6.2. Discussion

During the building definition, a function from the CGAL Polygon Mesh Processing package that identifies and splits connected components was used. This tool efficiently computes separate building objects, in a satisfactory manner. However, buildings are often defined based on individual solids, even though they may consist of multiple solids in the input files. Additionally, multi-surfaces were sometimes considered as buildings, resulting in buildings without volumes. While this does not influence the CFD simulations themselves, it can affect the statistics presented to users (e.g. number of overlapping buildings).

Thresholds were used to identify sliver triangles, short edges, and sharp angles based on default or user-defined values. However, their optimal values remain unknown. In other words, it is unclear at which dimensions these geometries pose a problem for the meshing process. Further research is needed to determine the optimal values for these thresholds. Moreover, their values are likely to depend on the input model and purpose of the CFD simulation, which also makes it difficult for users to choose the most appropriate values. For example, a higher Level of Detail (LoD), which often leads to more geometric details and smaller geometries, can lead to better simulation results (García-Sánchez et al., 2021). Therefore, smaller threshold values would be more suitable. Yet, this might also lead to lower mesh quality (and simulation results) and a more computationally expensive simulation. So, at present, choosing appropriate threshold values is based on experience with CFD simulation and thus still requires some expertise.

The topological relationships validation algorithm seems to work properly. However, it is strongly dependent on the method that defines ground surfaces of buildings. Building surfaces lower than a given threshold z and with an angle smaller than a threshold θ with regard to the ground are considered as ground surfaces. This approach might lead to incorrect results when, for example, buildings contain internal geometries, or have heights lower than z and flat roofs.

A selection of CFD guidelines was made and implemented in the mesh parameters definition. The following observations were done, based on validations presented in Section 5.2:

- The models were correctly rotated based on the input flow.
- The minimum requirements for the domain dimensions and optional blockage ratios seem to be respected. The blockage ratios result in very large domains, especially in the y-direction. This might affect the simulation results, as guidelines suggest larger dimensions in the x-direction (direction of the incoming flow). Also, larger domains lead to more computationally expensive simulations, which is not beneficial.
- The prototype successfully follows the roughness height z_0: the first evaluation node from the ground is always at a minimum of z_0.
- The target evaluation height was respected as long as the roughness height condition can be satisfied, which is as expected.
- The number of cells within the mesh is approximated and differs on average by -12.12% with the mesh generated in OpenFOAM. These differences vary between 3 to 38% (absolute values). It was difficult to determine a more accurate approximation due to certain tasks applied by OpenFOAM to optimize the mesh, which require further effort to understand.

- The identification of buildings with insufficient cells at each side (per cube root building volume) or per building separation appears to work as expected. One limitation here is the computation of building separations, which are considered as the distance between a vertex of the convex hull from Building A and one from building B. This method can be accurate (Figure 6.1 a), but there are some cases in which it is not (Figure 6.1 b). Additionally, the test cases in Sections 5.2.6 and 5.2.7 demonstrate that the requirements for the minimum number of cells per building and separation are rarely met. Yet, satisfying all CFD guidelines is nearly impossible, so this was to be expected. These identifications give only an indication of the extent to which the guidelines are followed.

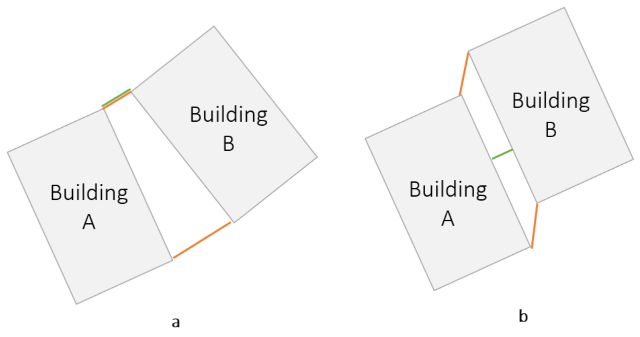


Figure 6.1.: Example of building separations: (a) accurate (b) inaccurate

- The Region of Interest (RoI) is correctly defined based on the guidelines of Liu et al. (2018).
- The ground refinement was shown to be correctly applied. Yet, the number of cell approximation differs by -22.2% with the final mesh generated in OpenFOAM, which is a larger difference than the approximation without ground refinement. Additionally, the number of cells approximation for ground surfaces is based on the perimeter of buildings approximated using their convex hull, which is less accurate than using their actual perimeters.

The CFD guidelines themselves, however, cannot all be satisfied at the same time. This is not necessarily a limit of the prototype but more of the guidelines itself.

Errors, warnings, and results are provided to users in the form of reports and visualisations. The prototype contributes to the simplification of using CFD simulations for urban areas. However, users still need sufficient expertise to set suitable input parameters for geometric validation and mesh parameters definitions. To help users with choosing adequate parameters, explanation is provided for each of these within the prototype.

The prototype helps users with preparing their geometries and defining pre-run setups for CFD simulations in urban areas, which are time consuming and prone to errors. This was achieved by identifying geometries that might affect the CFD meshing process and compute mesh parameters for OpenFOAM based on CFD guidelines. The used methods for these tasks had led to satisfactory results. This prototype is a small step towards making realistic urban CFD simulations available for everyone, which would be beneficial as issues related to flows in residential areas are likely to increase.

6.3. Recommendations

As described in the discussion (Section 6.2), the dimensions at which sliver triangles, sharp angles, and short edges may affect the CFD mesh are unknown. Currently, users have to choose these threshold values based on their experience. It would be useful to investigate their optimal values, and whether they depend on input models and/or simulation parameters (e.g. types of mesh).

The prototype could be improved with a more robust algorithm that identifies ground surfaces. The used method relies on two thresholds: one for the maximum height of ground surfaces, and one for the angle between ground and terrain surfaces (Appendix B). It would be useful to add an algorithm able to identify them without these indications, as reducing the preparation tasks is one of the objectives. Removing some parameters could contribute to this.

The observed difference of 12.12% in the number of cell approximations in a mesh without ground refinement and 22.22% with ground refinement from the actual number requires further investigation. These discrepancies appear to be due to optimisation tasks performed by OpenFOAM during the meshing process, which require thorough investigation to be understood.

As mentioned earlier, some CFD guidelines can not be respected at the same time. Further work could explore guidelines that are both achievable and lead to realistic results in urban areas.

The prototype can further be developed to contribute to simplify the use of CFD simulations in urban areas. Some ideas include implementing boundary conditions of the domain and expanding the range of domain types (e.g. round and oval). Currently, only rectangle domains are considered.

A. Topological relationships validations

In this appendix, some more validations are showed for the topological relationships validations, discussed in Section 5.1.1.

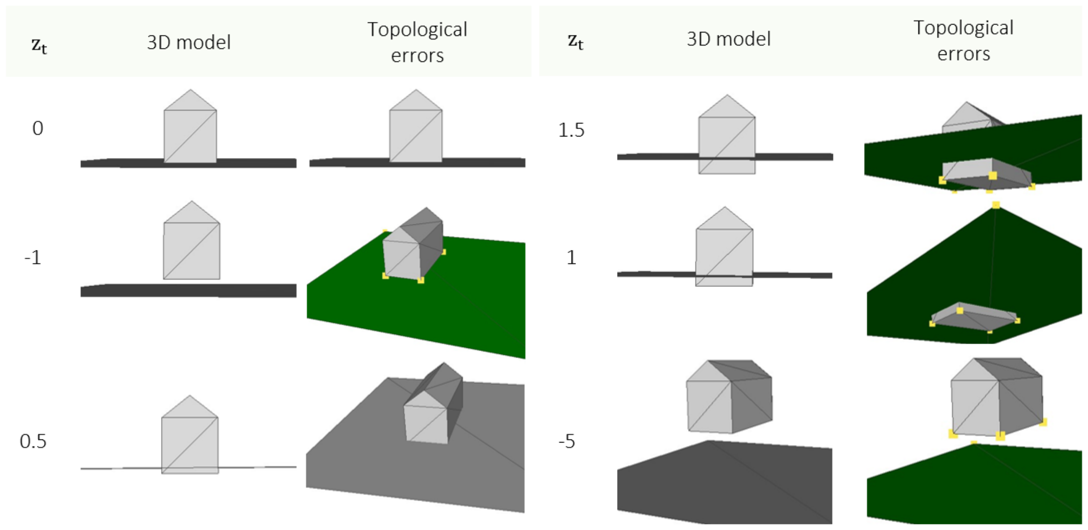


Figure A.1.: Testing topological relationships validation with single building models, having horizontal floors with a height of 0, and horizontal terrain surfaces with a height of z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

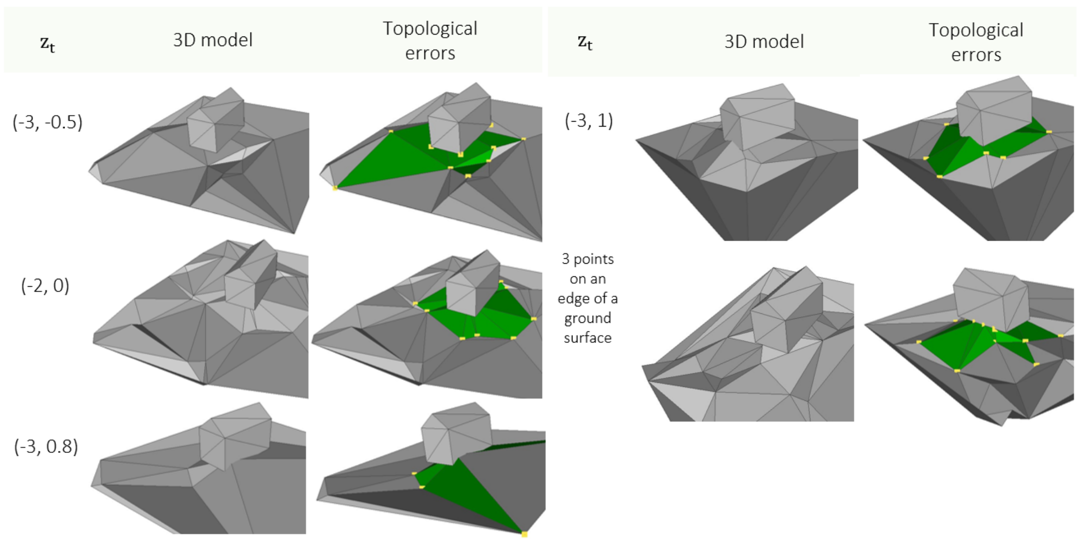


Figure A.2.: Testing topological relationships validation with single building models, having horizontal floors with a height of 0, and terrain surfaces with variable heights within a range z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

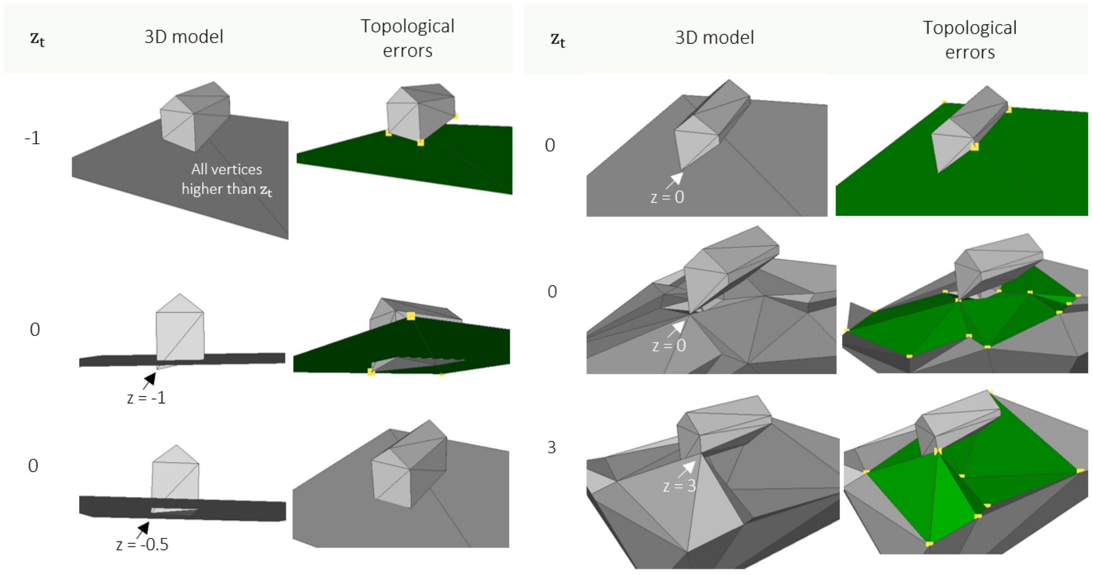


Figure A.3.: Testing topological relationships validation with single building models, having floors with variable heights z, and terrain surfaces with variable heights z_t. Threshold value ζ is set to 1. Terrain surfaces and building vertices forming topological errors are highlighted in green and yellow, respectively, as they are incorrectly positioned in relation to each other (Section 3.2.2).

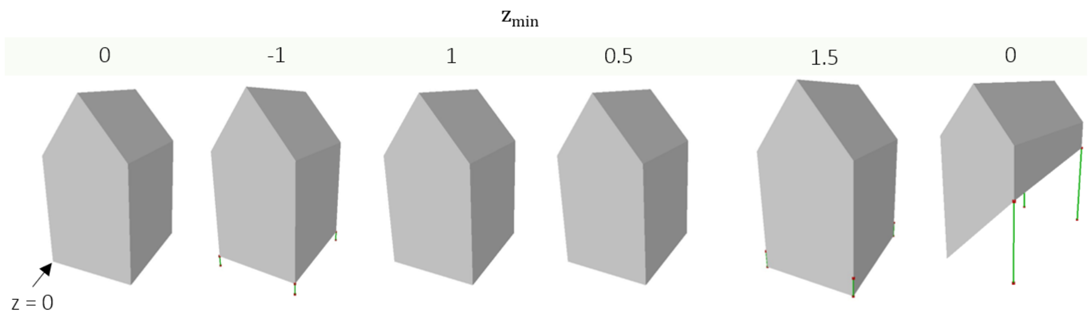


Figure A.4.: Testing topological relationships validation with single buildings and without ground surfaces. z_t is the height at which buildings must be placed. The lowest z-value of the buildings (z_{min}) is 0. Threshold value ζ is set to 1.

B. Ground surfaces

Algorithm B.1 defines ground surfaces. For each vertex v of every face f that forms a building b, the z coordinates z_1, z_2, and z_3 were retrieved and compared to a threshold height h (default value: 3). If all these z coordinates are lower or equal to h, the normal vector n (x_n, y_n, z_n) of f is computed. Then, θ_x and θ_y are calculated (Figure B.1), which are the angles between vectors (0, 0, -1) and (x_n, 0, -1) and between vectors (0, 0, -1) and (0, y_n, -1), respectively. If these angles are lower than a threshold angle θ (default: 45°) and z_n is lower than 0°, f is considered as a ground surface.

Algorithm B.1: Ground surfaces.

```
Data: Buildings $B$ forming of faces $f_b$, height $h$, and threshold angle $\theta$
Result: Ground surfaces $gs$
for all $b \in B$ do
    for all $f \in f_b$ do
        $z_1 \leftarrow$ z coordinate of vertex 1;
        $z_2 \leftarrow$ z coordinate of vertex 2;
        $z_3 \leftarrow$ z coordinate of vertex 3;
        if $z_1 \leq h$ and $z_2 \leq h$ and $z_3 \leq h$ then
            $n \leftarrow$ normal vector of $f$;
            $x_n \leftarrow$ x coordinate of $n$;
            $y_n \leftarrow$ y coordinate of $n$;
            $z_n \leftarrow$ z coordinate of $n$;
            $\theta_x \leftarrow abs(\arctan \frac{x_n}{z_n}) \cdot \frac{180}{\pi}$;
            $\theta_y \leftarrow abs(\arctan \frac{y_n}{z_n}) \cdot \frac{180}{\pi}$;
            if $\theta_x \leq \theta$ and $\theta_y \leq \theta$ and $z_n < 0$ then
                label $f$ as ground surface $gs$;
            else
                label $f$ as not a ground surface;
```

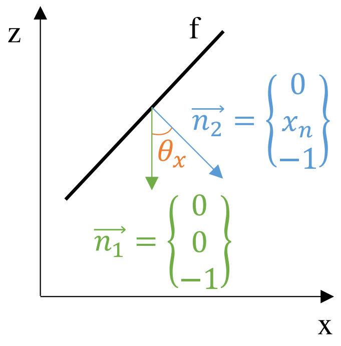

z
f
$\overrightarrow{n_2} = \begin{Bmatrix} 0 \\ x_n \\ -1 \end{Bmatrix}$
θ_x
$\overrightarrow{n_1} = \begin{Bmatrix} 0 \\ 0 \\ -1 \end{Bmatrix}$
x

Figure B.1.: Normal angles computed for a ground surface f.

The ground surfaces algorithm was validated by using single building models with different ground surfaces and different threshold values h and θ. First, single building models with two ground surfaces having the same slopes were tested. The results are illustrated in Figure B.2. Then, single building models were tested with two ground surfaces having different slopes. Figure B.3 shows the results. It demonstrates that Algorithm B.1 verifies both angles θ_x and θ_y. For one of the surfaces in the second case, the normal of the first face N_1 has a θ_x value of 50 °, and θ_y value of 45 °. As θ_x is greater than the threshold value θ, it is not considered as a ground surface. The ground surfaces are highlighted in green in both figures.

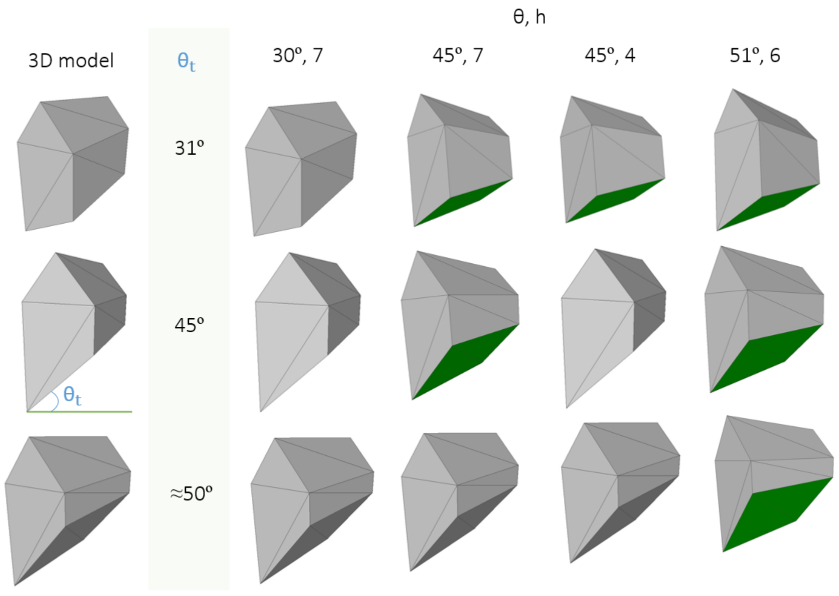


Figure B.2.: Ground surfaces identification for single buildings with ground surfaces having the same slope. θ_t represents their angles with the terrain. Surfaces are selected that have an angle with regard to the terrain lower than θ, and have all their vertices lower than threshold value h. Ground surfaces are highlighted in green.

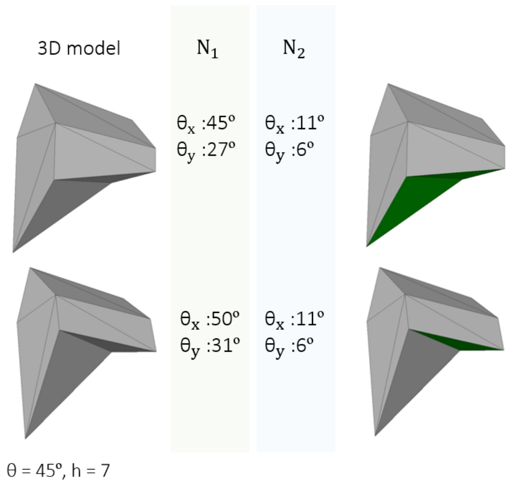


Figure B.3.: Ground surfaces identification for single buildings with ground surfaces with different slopes. Each building has two ground surfaces with normals N_1 and N_2. θ_x is the angle between vectors (0, 0, -1) and (x_n, 0, -1) as represented in Figure B.1, and θ_y is the same angle but then in the y-direction, resulting in the angle between (0, 0, -1) and (0, y_n, -1).

C. Sharp angles validations

In this appendix, some more validations are showed for the sharp angles identification, discussed in Section 5.1.2.

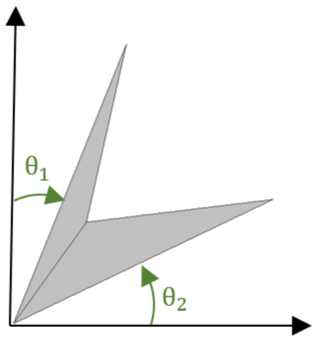


Figure C.1.: Two connected triangles.

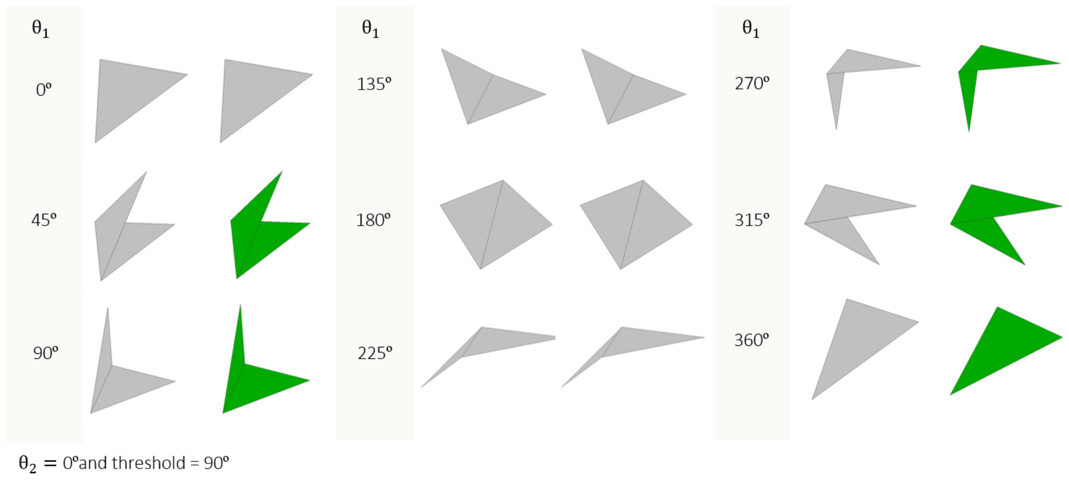
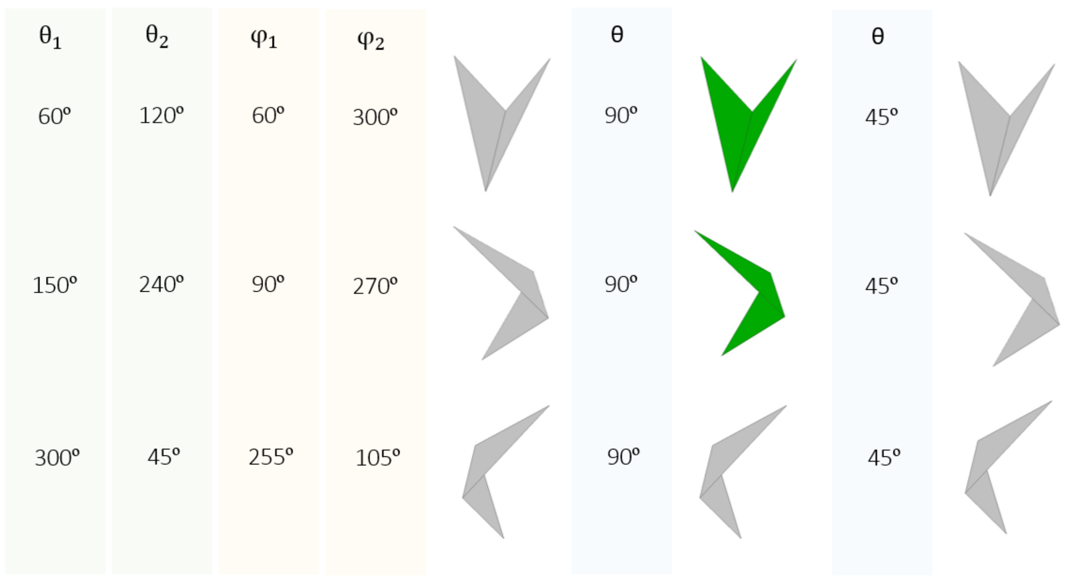
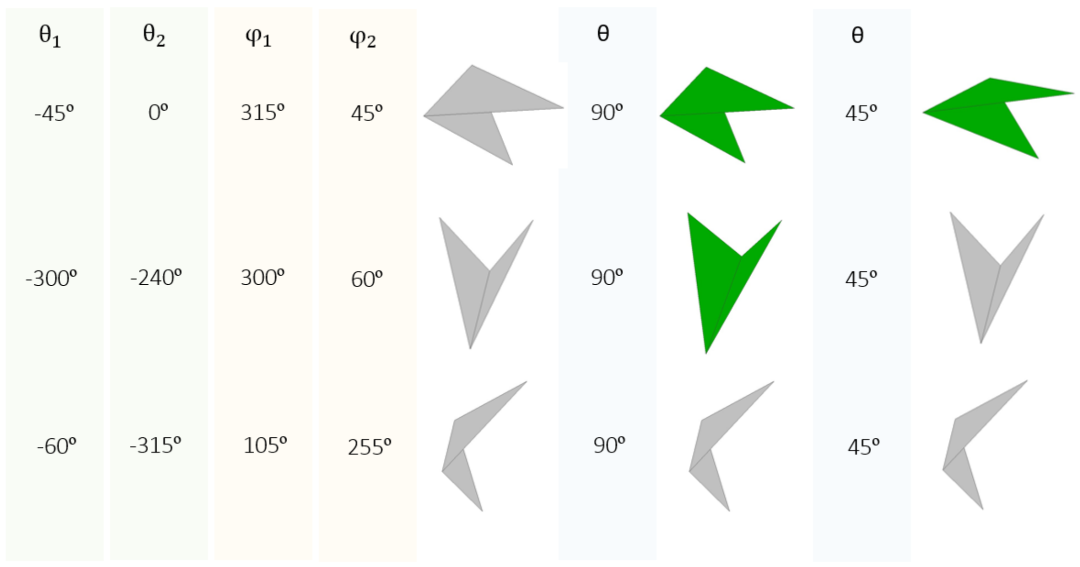


Figure C.2.: Sharp angles identification of two connected triangles with a threshold value θ of 90°. θ_1 and θ_2 are the angles illustrated in Figure C.1. Only one of the two triangles varies in angle (θ_1), the other one remains at 0°. The threshold θ is set to 90○. Sharp angles are indicated in green.



θ_1	θ_2	φ_1	φ_2	θ	θ
60°	120°	60°	300°	90°	45°
150°	240°	90°	270°	90°	45°
300°	45°	255°	105°	90°	45°

Figure C.3.: Sharp angles identification of two connected triangles with a threshold value θ of 90° and 45°. θ_1 and θ_2 are the angles illustrated in Figure C.1. ϕ_1 is the first angle between these triangles and ϕ_2 is the second one. Sharp angles are indicated in green.



θ_1	θ_2	φ_1	φ_2	θ	θ
-45°	0°	315°	45°	90°	45°
-300°	-240°	300°	60°	90°	45°
-60°	-315°	105°	255°	90°	45°

Figure C.4.: Same validations performed as explained Figure C.3, but then with negative angles.

D. Sliver triangles validations

In this appendix, some more validations are showed for the sliver triangles identification, discussed in Section 5.1.2.

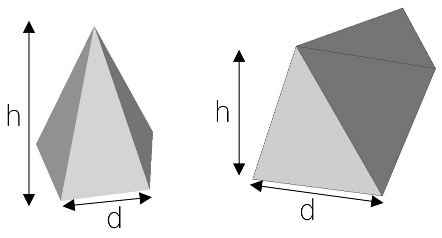


Figure D.1.: Cone and roof models.

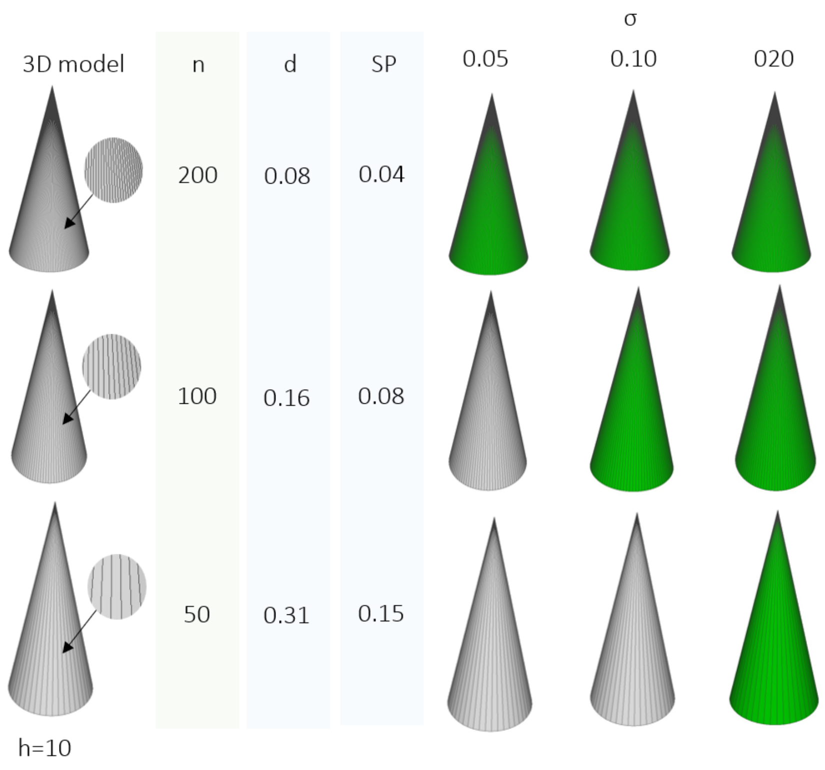
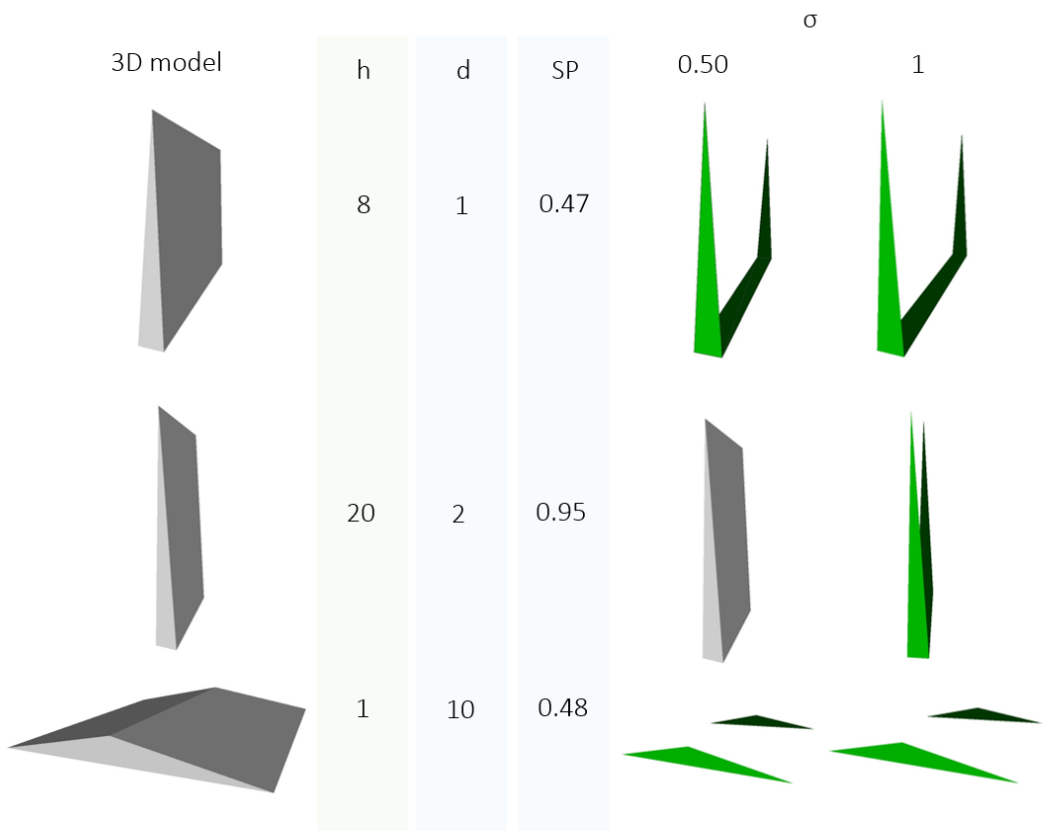


Figure D.2.: Sliver triangles identification with cones (n: number of triangles, d and h: distances as illustrated in Figure D.1, and SP: sliver parameter), and different threshold values σ for the sliver parameter. Sliver triangles are indicated in green.



h	d	SP
8	1	0.47
20	2	0.95
1	10	0.48

Figure D.3.: Sliver triangles identification of roofs (n: number of triangles, d and h: distances as illustrated in Figure D.1, and SP: sliver parameter), and different threshold values σ for the sliver parameter. Sliver triangles are indicated in green.

E. Short edges validations

In this appendix, some more validations are showed for the short edges identification, discussed in Section 5.1.2.

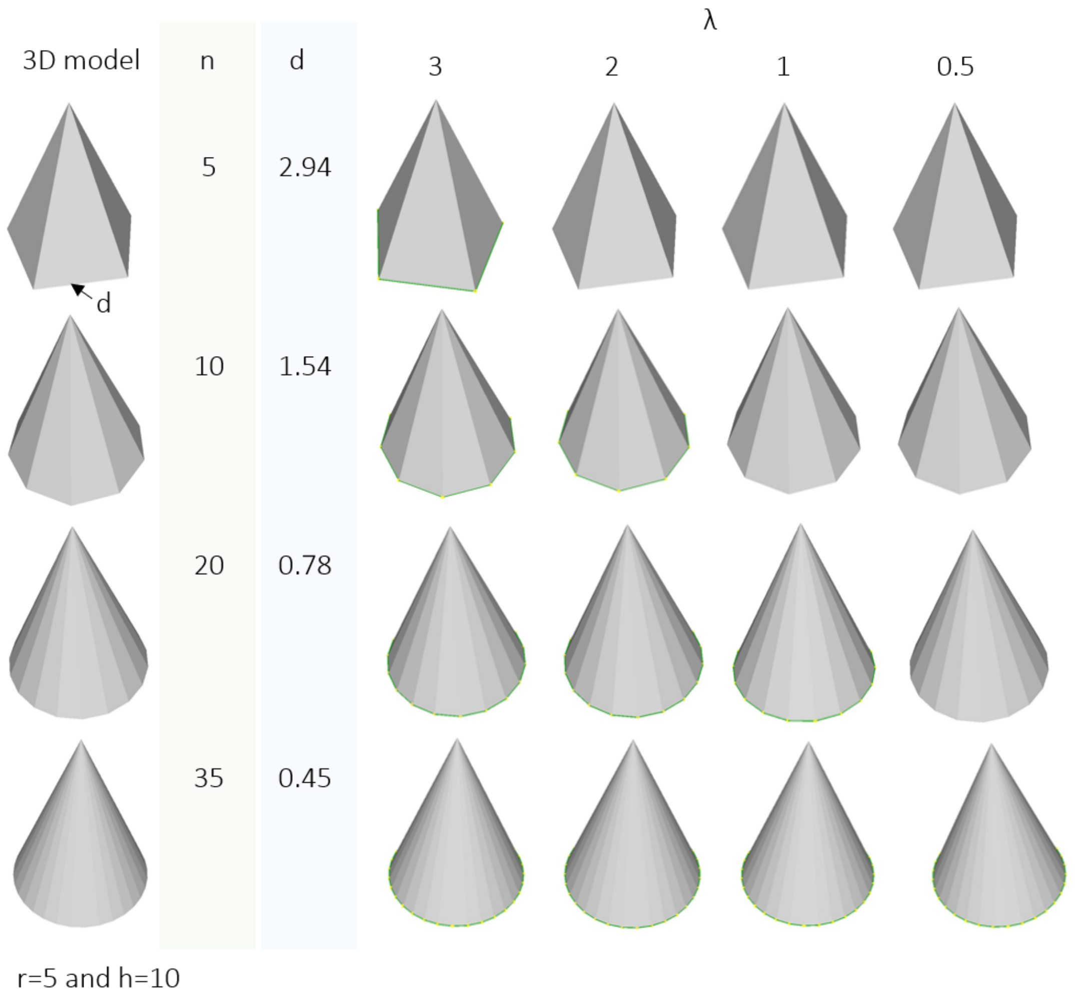


Figure E.1.: Short edges identification of cones (n: number of triangles, h: height, and r: radius), and different threshold values for the length of edges. Short edges are indicated in green.

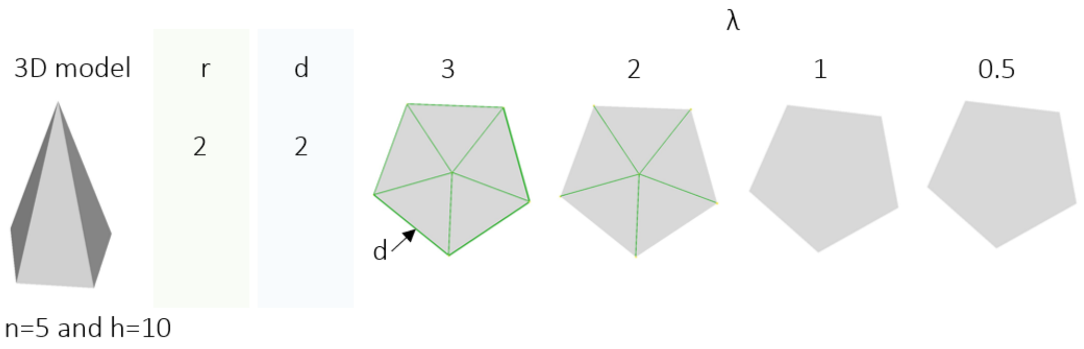


Figure E.2.: Short edges identification of cones (n: number of triangles, h: height, and r: radius), and different threshold values for the length of edges. Short edges are indicated in green.

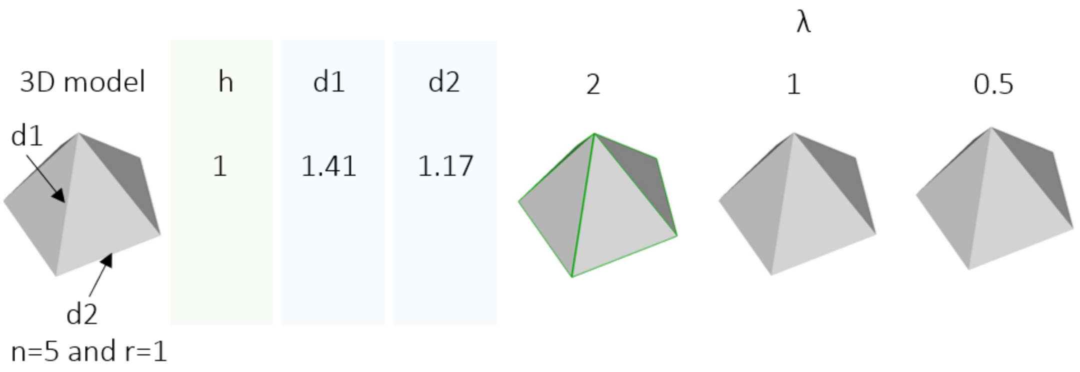
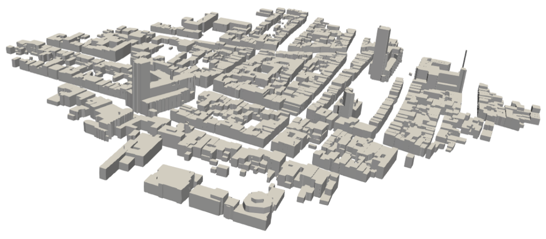


Figure E.3.: Short edges identification of cones (n: number of triangles, h: height, and r: radius), and different threshold values for the length of edges. Short edges are indicated in green.

F. CFD tests

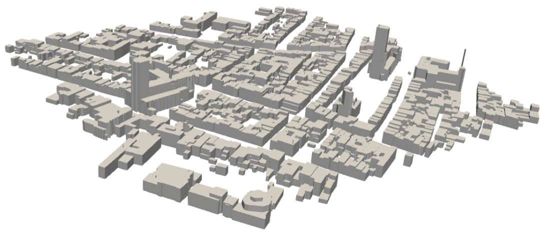
F.0.1. Evaluation height h_{user}

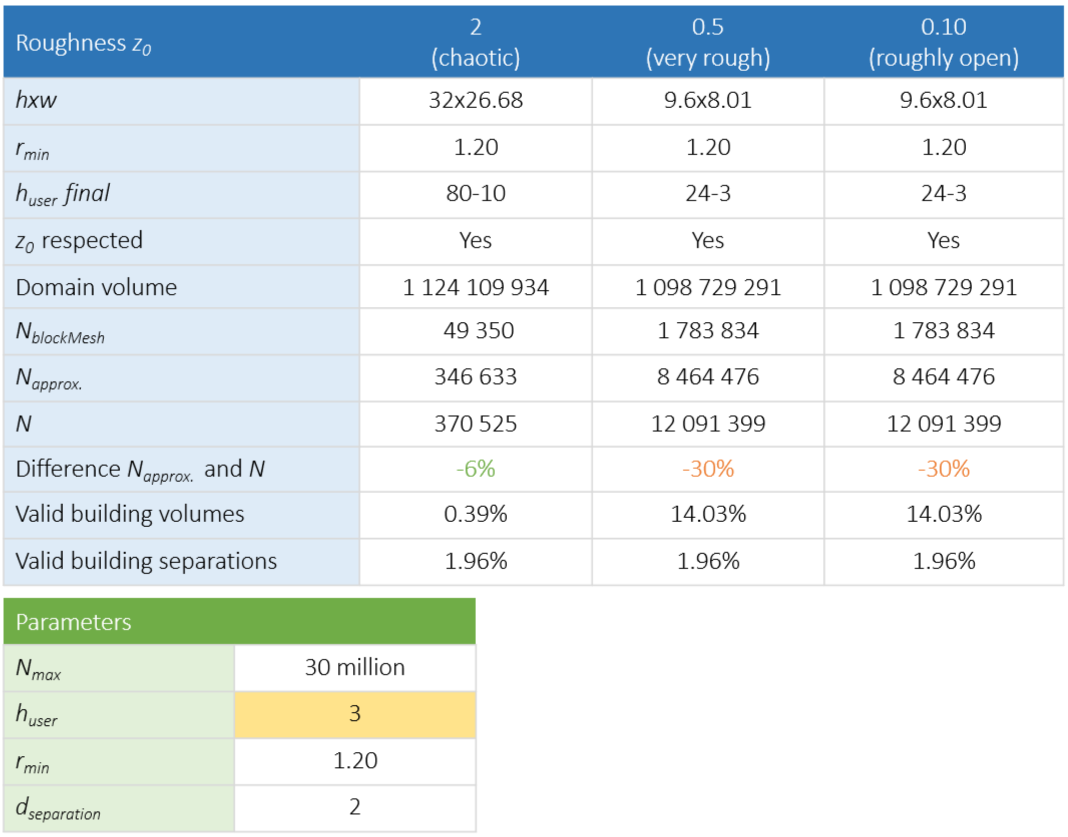


Roughness z_0	2 (chaotic)	0.5 (very rough)	0.10 (roughly open)
hxw	32x26.68	8x6.68	5.4x5.39
r_{min}	1.20	1.20	1.00
h_{user} *final*	80-10	20-2.5	13.5-1.69
z_0 respected	Yes	Yes	Yes
Domain volume	1 124 109 934	1 092 584 819	1 081 957 890
$N_{blockMesh}$	49 350	3 060 640	6 896 664
$N_{approx.}$	346 633	13 629 727	29 515 237
N	370 525	17 601 037	44 194 102
Difference $N_{approx.}$ and N	-6%	-22%	-33%
Valid building volumes	0.39%	30.32%	76.74%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	1.50
r_{min}	1.20
$d_{separation}$	2

Figure F.1.: Mesh parameters suggested by the prototype for the Centre of Delft with an evaluation height h_{user} of 1.5m.

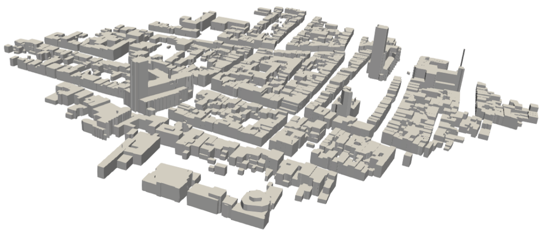




Roughness z_0	2 (chaotic)	0.5 (very rough)	0.10 (roughly open)
hxw	32x26.68	9.6x8.01	9.6x8.01
r_{min}	1.20	1.20	1.20
h_{user} *final*	80-10	24-3	24-3
z_0 respected	Yes	Yes	Yes
Domain volume	1 124 109 934	1 098 729 291	1 098 729 291
$N_{blockMesh}$	49 350	1 783 834	1 783 834
$N_{approx.}$	346 633	8 464 476	8 464 476
N	370 525	12 091 399	12 091 399
Difference $N_{approx.}$ and N	-6%	-30%	-30%
Valid building volumes	0.39%	14.03%	14.03%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	3
r_{min}	1.20
$d_{separation}$	2

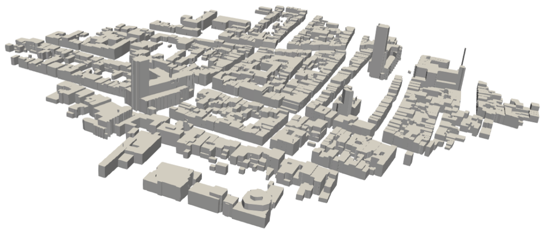
Figure F.2.: Mesh parameters suggested by the prototype for the Centre of Delft with an evaluation height h_{user} of 3m.



Roughness z_0	2 (chaotic)	0.5 (very rough)	0.10 (roughly open)
hxw	32x26.68	16x13.35	16x13.35
r_{min}	1.20	1.20	1.20
h_{user} *final*	80-10	40-5	40-5
z_0 respected	Yes	Yes	Yes
Domain volume	1 124 109 934	1 118 290 582	1 118 290 582
$N_{blockMesh}$	49 350	392 168	392 168
$N_{approx.}$	346 633	2 128 997	2 128 997
N	370 525	2 845 849	2 845 849
Difference $N_{approx.}$ and N	-6%	-25%	-25%
Valid building volumes	0.39%	1.67%	1.67%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	5
r_{min}	1.20
$d_{separation}$	2

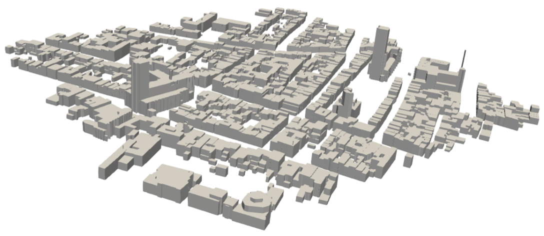
Figure F.3.: Mesh parameters suggested by the prototype for the Centre of Delft with an evaluation height h_{user} of 5m.



Roughness z_0	2 (chaotic)	0.5 (very rough)	0.10 (roughly open)
hxw	32x26.68	32x26.68	32x26.68
r_{min}	1.20	1.20	1.20
h_{user} *final*	80-10	80-10	80-10
z_0 respected	Yes	Yes	Yes
Domain volume	1 124 109 934	1 124 109 934	1 124 109 934
$N_{blockMesh}$	49 350	49 350	49 350
$N_{approx.}$	346 633	346 633	346 633
N	370 525	370 525	370 525
Difference $N_{approx.}$ and N	-6%	-6%	-6%
Valid building volumes	0.39%	0.39%	0.39%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	10
r_{min}	1.20
$d_{separation}$	2

Figure F.4.: Mesh parameters suggested by the prototype for the Centre of Delft with an evaluation height h_{user} of 10m.

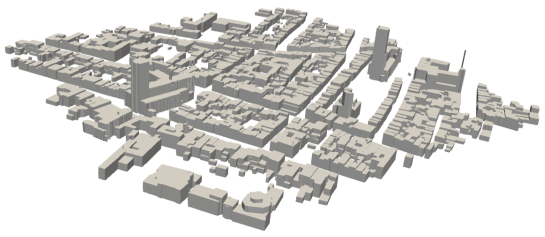


Roughness z_0	2 (chaotic)	0.5 (very rough)	0.10 (roughly open)
hxw	64x53.35	64x53.35	64x53.35
r_{min}	1.20	1.20	1.20
h_{user} *final*	160-20	160-20	160-20
z_0 respected	Yes	Yes	Yes
Domain volume	1 162 898 204	1 162 898 204	1 162 898 204
$N_{blockMesh}$	6 384	6 384	6 384
$N_{approx.}$	67 023	67 023	67 023
N	69 576	69 576	69 576
Difference $N_{approx.}$ and N	-4%	-4%	-4%
Valid building volumes	0.19%	0.19%	0.19%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	20
r_{min}	1.20
$d_{separation}$	2

Figure F.5.: Mesh parameters suggested by the prototype for the Centre of Delft with an evaluation height h_{user} of 20m.

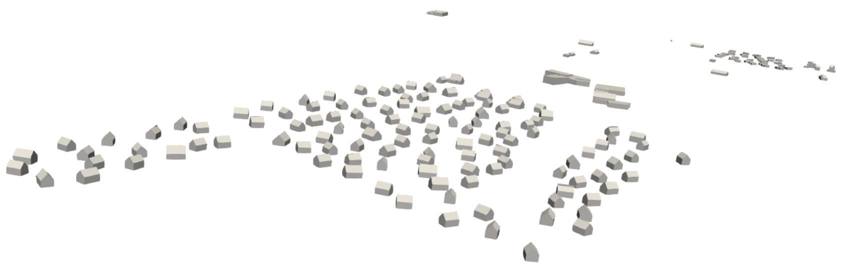
F.0.2. Roughness height z_0



Roughness z_0	2 (chaotic)	1 (closed)	0.5 (very rough)	0.25 (rough)	0.10 (roughly open)	0.03 (open)
hxw	32x26.68	16x13.35	8x6.68	5.4x5.39	5.4x5.39	5.4x5.39
r_{min}	1.20	1.20	1.20	1.00	1.00	1.00
h_{user} *final*	80-10	40-5	20-2.5	13.5-1.69	13.5-1.69	13.5-1.69
z_0 respected	Yes	Yes	Yes	Yes	Yes	Yes
Domain volume	1 124 109 934	1 118 290 582	1 092 584 819	1 081 957 890	1 081 957 890	1 081 957 890
$N_{blockMesh}$	49 350	392 168	3 060 640	6 896 664	6 896 664	6 896 664
$N_{approx.}$	346 633	2 128 997	13 629 727	29 515 237	29 515 237	29 515 237
N	370 525	2 486 196	17 601 037	44 194 102	44 194 102	44 194 102
Difference $N_{approx.}$ and N	-6%	-14%	-22%	-33%	-33%	-33%
Valid building volumes	0.39%	1.67%	30.32%	76.74%	76.74%	76.74%
Valid building separations	1.96%	1.96%	1.96%	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	1.50
r_{min}	1.20
$d_{separation}$	2

Figure F.6.: Mesh parameters suggested by the prototype for the Centre of Delft with different roughness height values z_0.

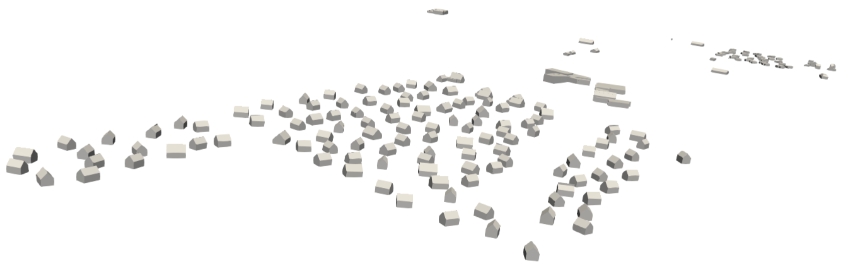


Roughness z_0	2 (chaotic)	1 (closed)	0.5 (very rough)	0.25 (rough)	0.10 (roughly open)	**0.03 (open)**
hxw	32x26.68	16x13.35	8x6.68	4.8x4.01	4.8x4.01	4.8x4.01
r_{min}	1.20	1.20	1.20	1.20	1.20	1.20
h_{user} *final*	80-10	40-5	20-2.5	12-1.5	12-1.5	12-1.5
z_0 respected	Yes	Yes	Yes	Yes	Yes	Yes
Domain volume	61 501 455	59 848 541	59 938 236	53 794 495	53 794 495	53 794 495
$N_{blockMesh}$	2 700	20 988	167 904	696 960	696 960	696 960
$N_{approx.}$	89 468	652 433	4 973 599	20 575 779	20 575 779	20 575 779
N	130 848	715 050	4 766 718	18 841 020	18 841 020	18 841 020
Difference $N_{approx.}$ and N	-32%	-9%	4%	9%	9%	9%
Valid building volumes	0%	1.18%	4.12%	80.59%	80.59%	80.59%
Valid building separations	0.59%	0.59%	0.59%	18.82%	18.82%	18.82%

Parameters	
N_{max}	30 million
h_{user}	1.50
r_{min}	1.20
$d_{separation}$	2

Figure F.7.: Mesh parameters suggested by the prototype for Ouddorp with different roughness height values z_0.

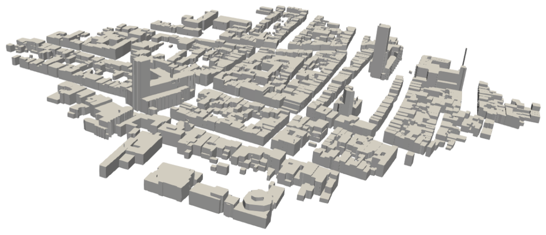
F.0.3. Maximum number of cells N_{max}



N_{max}	1 000	10^6	10 x 10^6	30 x 10^6	50 x 10^6	80 x 10^6	100 x 10^6
hxw	285.7x285.69	12.1x11.3	8x6.68	8x6.68	8x6.68	8x6.68	8x6.68
r_{min}	1.00	1.07	1.20	1.20	1.20	1.20	1.20
h_{user} *final*	714.25-89.28	30.25-3.78	20-2.50	20-2.5	20-2.50	20-2.50	20-2.50
z_0 respected	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Domain volume	349 777 265	56 942 781	59 938 236	59 938 236	59 938 236	59 938 236	59 938 236
$N_{blockMesh}$	15	36 855	167 904	167 904	167 904	167 904	167 904
$N_{approx.}$	933	995 744	4 973 599	4 973 599	4 973 599	4 973 599	4 973 599
N	792	1 037 694	4 766 718	4 766 718	4 766 718	4 766 718	4 766 718
Difference $N_{approx.}$ and N	18%	-4%	4%	4%	4%	4%	4%
Valid building volumes	0%	2.35%	4.12%	4.12%	4.12%	4.12%	4.12%
Valid building separations	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%	0.59%

Parameters	
z_0	0.50
h_{user}	1.50
r_{min}	1.20
$d_{separation}$	2

Figure F.8.: Mesh parameters suggested by the prototype for Ouddorp with different number of cell limits values N_{max}.

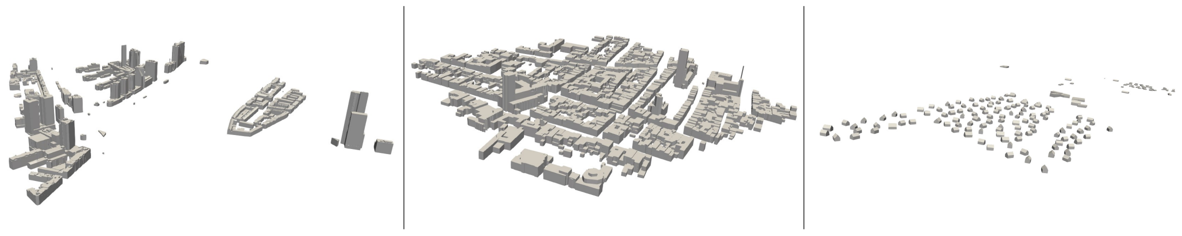


N_{max}	1 000	10^6	10 x 10^6	30 x 10^6	50 x 10^6	80 x 10^6	100 x 10^6
hxw	432.3x432.26	19x19	8x7.93	8x6.68	8x6.68	8x6.68	8x6.68
r_{min}	1.00	1.00	1.01	1.20	1.20	1.20	1.20
h_{user} *final*	1080.75-135.09	47.5-5.94	20-2.5	20-2.5	20-2.5	20-2.5	20-2.5
z_0 respected	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Domain volume	2 423 240 889	1 082 844 048	1 092 939 562	1 092 584 819	1 092 584 819	1 092 584 819	1 092 584 819
$N_{blockMesh}$	30	157 872	2 172 500	3 060 640	3 060 640	3 060 640	3 060 640
$N_{approx.}$	1000	981 008	9 999 060	13 629 727	13 629 727	13 629 727	13 629 727
N	1 436	1 243 126	14 464 334	17 601 037	17 601 037	17 601 037	17 601 037
Difference $N_{approx.}$ and N	-30%	-21%	-31%	-22%	-22%	-22%	-22%
Valid building volumes	0%	1.08%	30.32%	30.32%	30.32%	30.32%	30.32%
Valid building separations	1.96%	1.96%	1.96%	1.96%	1.96%	1.96%	1.96%

Parameters	
z_0	0.50
h_{user}	1.50
r_{min}	1.20
$d_{separation}$	2

Figure F.9.: Mesh parameters suggested by the prototype for the Centre of Delft with different number of cell limits values N_{max}.

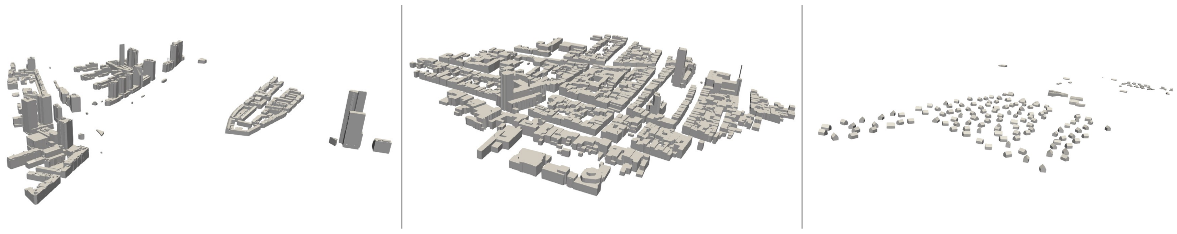
F.0.4. Height of the tallest building h_{max}



Model	Maastoren (h_{max} = 168.05)	Delft (h_{max} = 89.42)	Ouddorp (h_{max} = 9.40)
hxw	12.9x12.84	5.4x5.39	4.8x4.01
r_{min}	1.00	1.00	1.20
h_{user} *final*	32.25-4.03	13.5-1.69	12-1.5
z_0 respected	Yes	Yes	Yes
Domain volume	12 426 354 858	1 081 957 890	53 794 495
$N_{blockMesh}$	5 842 840	6 896 664	696 960
$N_{approx.}$	29 643 109	29 515 237	20 575 779
N	31 203 522	44 194 102	18 841 020
Difference $N_{approx.}$ and N	-5%	-33%	9%
Valid building volumes	15.26%	76.74%	80.59%
Valid building separations	9.93%	1.96%	18.82%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.25
r_{min}	1.20
$d_{separation}$	2

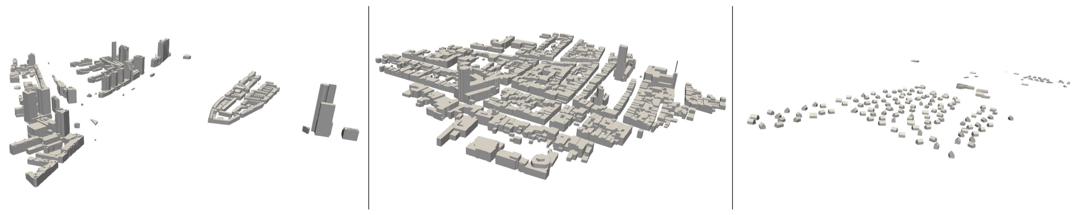
Figure F.10.: Mesh parameters suggested by the prototype for Maastoren, the Centre of Delft, and Ouddorp with a roughness height value z_0 of 0.25m.



Model	Maastoren (h_{max} = 168.05)	Delft (h_{max} = 89.42)	Ouddorp (h_{max} = 9.40)
hxw	12.9x12.84	8x6.68	8x6.68
r_{min}	1.00	1.20	1.20
h_{user} *final*	32.25-4.03	20-2.5	20-2.5
z_0 respected	Yes	Yes	Yes
Domain volume	12 426 354 858	1 092 584 819	59 938 236
$N_{blockMesh}$	5 842 840	3 060 640	167 904
$N_{approx.}$	29 643 109	13 629 727	4 973 599
N	31 203 522	17 601 037	4 766 718
Difference $N_{approx.}$ and N	-5%	-22%	4%
Valid building volumes	15.26%	30.32%	4.12%
Valid building separations	9.93%	1.96%	0.59%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.5
r_{min}	1.20
$d_{separation}$	2

Figure F.11.: Mesh parameters suggested by the prototype for Maastoren, the Centre of Delft, and Ouddorp with a roughness height value z_0 of 0.5m.

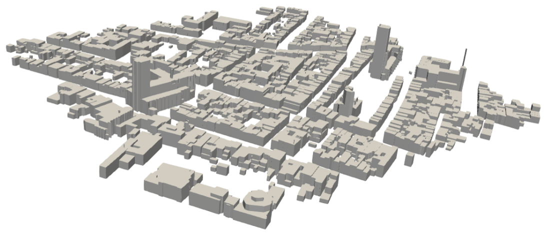


Model	Maastoren (h_{max} = 168.05)	Delft (h_{max} = 89.42)	Ouddorp (h_{max} = 9.40)
hxw	12.9x12.84	5.4x5.39	4.8x4.01
r_{min}	1.00	1.00	1.20
h_{user} *final*	32.25-4.03	13.5-1.69	12-1.5
z_0 respected	Yes	Yes	Yes
Domain volume	12 426 354 858	1 081 957 890	53 794 495
$N_{blockMesh}$	5 842 840	6 896 664	696 960
$N_{approx.}$	29 643 109	29 515 237	20 575 779
N	31 203 522	44 194 102	18 841 020
Difference $N_{approx.}$ and N	-5%	-33%	9%
Valid building volumes	15.26%	76.74%	80.59%
Valid building separations	9.93%	1.96%	18.82%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.25
r_{min}	1.20
$d_{separation}$	2

Figure F.12.: Mesh parameters suggested by the prototype for Maastoren, the Centre of Delft, and Ouddorp with a roughness height value z_0 of 2m.

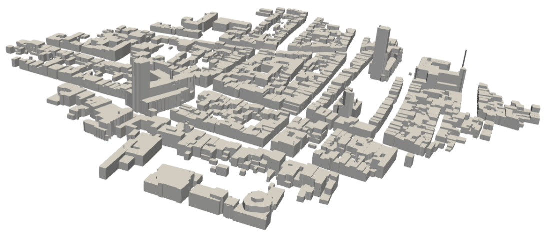
F.0.5. Ground refinement



	No GR	GR, no N_{max}	GR, N_{max}
hxw	5.4x5.39	5.4x5.39	6.2x6.16
r_{min}	1.00	1.00	1.01
h_{user} *final*	13.5-1.69	13.5-1.69	15.50-1.94
z_0 respected	Yes	Yes	Yes
Domain volume	1 081 957 890	1 081 957 890	1 088 460 148
$N_{blockMesh}$	6 896 664	6 896 664	4 626 573
$N_{approx.}$	29 515 237	40 503 050	29 736 376
N	44 194 102	65 682 877	40 976 798
Difference $N_{approx.}$ and N	-33%	-38%	-27%
Valid building volumes	76.74%	76.74%	65.06%
Valid building separations	1.96%	2.06%	1.96%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.25
r_{min}	1.20
$d_{separation}$	2

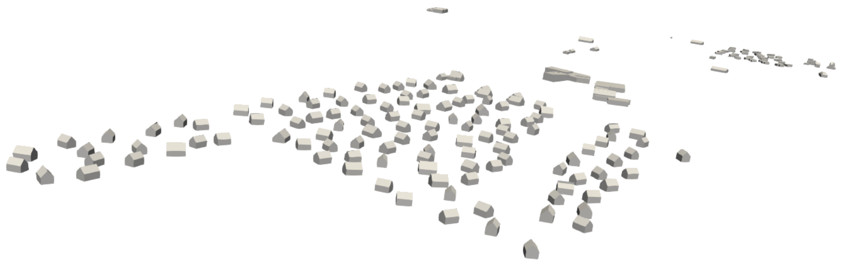
Figure F.13.: Mesh parameters suggested by the prototype for the Centre of Delft with a roughness height value z_0 of 0.25m when a ground refinement (GR) is applied with and without number of cells limit N_{max}.



	No GR	GR, no N_{max}	GR, N_{max}
hxw	8x6.68	8x6.68	8x6.68
r_{min}	1.20	1.20	1.20
h_{user} *final*	20-2.5	20-2.5	20-2.5
z_0 respected	Yes	Yes	Yes
Domain volume	1 092 584 819	1 092 584 819	1 092 584 819
$N_{blockMesh}$	3 060 640	3 060 640	3 060 640
$N_{approx.}$	13 629 727	24 685 982	24 685 982
N	17 601 037	32 639 187	32 639 187
Difference $N_{approx.}$ and N	-22%	-24%	-24%
Valid building volumes	30.32%	30.32%	30.32%
Valid building separations	1.96%	1.96%	1.96%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.5
r_{min}	1.20
$d_{separation}$	2

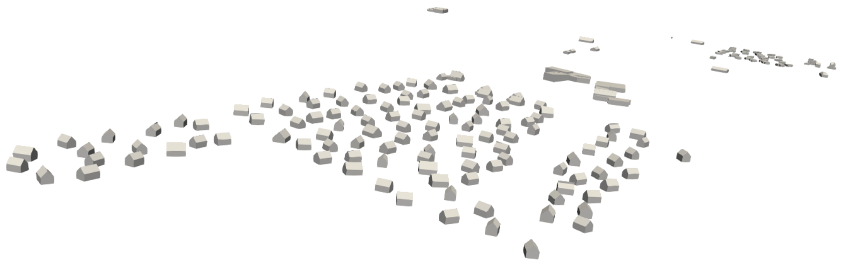
Figure F.14.: Mesh parameters suggested by the prototype for the Centre of Delft with a roughness height value z_0 of 0.5m when a ground refinement (GR) is applied with and without number of cells limit N_{max}.



	No GR	GR, no N_{max}	GR, N_{max}
hxw	4.8x4.01	4.8x4.01	4.8x4.01
r_{min}	1.20	1.20	1.20
h_{user} *final*	12-1.5	12-1.5	12-1.5
z_0 respected	Yes	Yes	Yes
Domain volume	53 794 495	53 794 495	53 794 495
$N_{blockMesh}$	696 960	696 960	696 960
$N_{approx.}$	20 575 779	22 467 191	22 467 191
N	18 841 020	29 792 968	29 792 968
Difference $N_{approx.}$ and N	9%	-25%	-25%
Valid building volumes	80.59%	80.59%	80.59%
Valid building separations	18.82%	21.18%	21.18%

Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.25
r_{min}	1.20
$d_{separation}$	2

Figure F.15.: Mesh parameters suggested by the prototype for Ouddorp with a roughness height value z_0 of 0.25m when a ground refinement (GR) is applied with and without number of cells limit N_{max}.



	No GR	GR, no N_{max}	GR, N_{max}
hxw	8x6.68	8x6.68	8x6.68
r_{min}	1.20	1.20	1.20
h_{user} *final*	20-2.5	20-2.5	20-2.5
z_0 respected	Yes	Yes	Yes
Domain volume	59 938 236	59 938 236	59 938 236
$N_{blockMesh}$	167 904	167 904	167 904
$N_{approx.}$	4 973 599	8 985 786	8 985 786
N	4 766 718	8 660 516	8 660 516
Difference $N_{approx.}$ and N	4%	3%	3%
Valid building volumes	4.12%	4.12%	4.12%
Valid building separations	0.59%	2.35%	2.35%

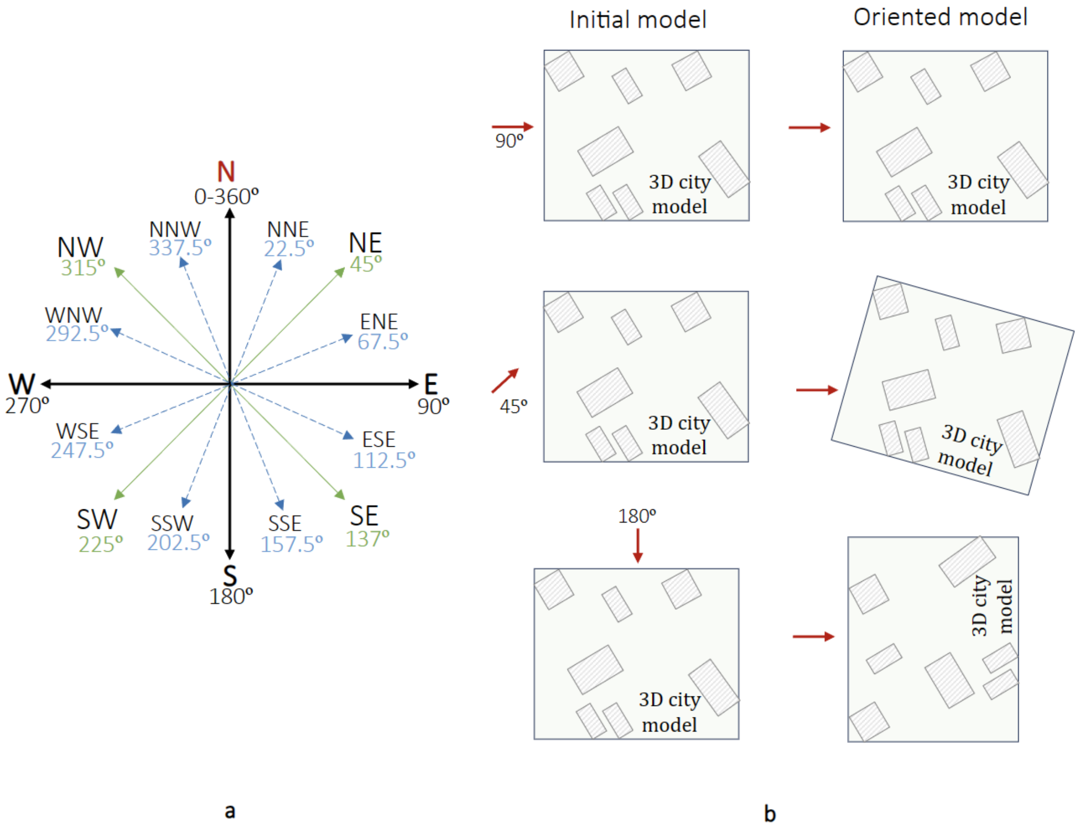
Parameters	
N_{max}	30 million
h_{user}	1.5
z_0	0.5
r_{min}	1.20
$d_{separation}$	2

Figure F.16.: Mesh parameters suggested by the prototype for Ouddorp with a roughness height value z_0 of 0.5m when a ground refinement (GR) is applied with and without number of cells limit N_{max}.

G. User Interface IV: explanation

G.1. Mesh definition parameters

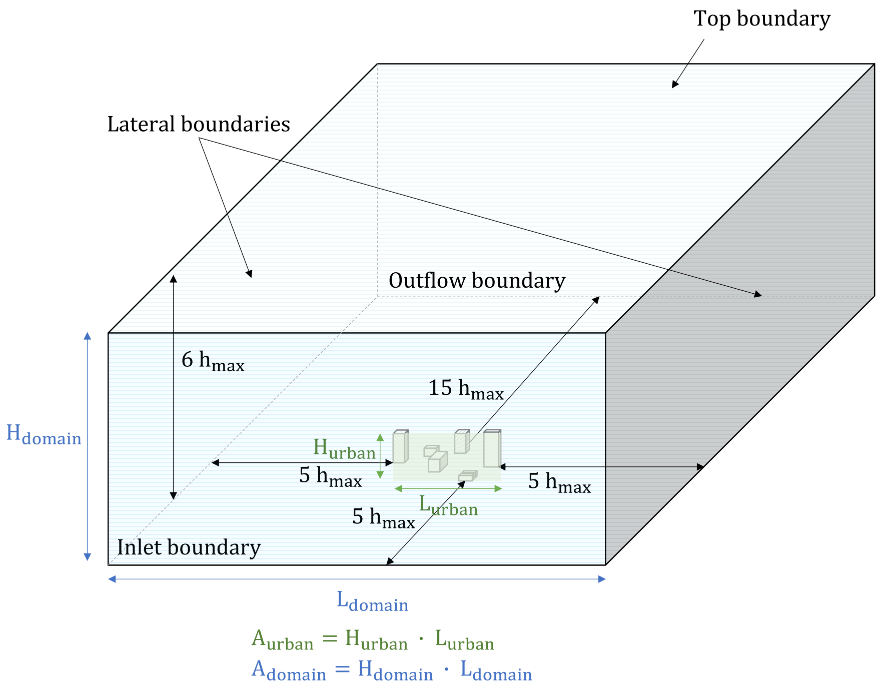
Flow direction
The 3D city model needs to be aligned with the incoming flow as simulated in OpenFOAM, which flows along the x-axis. The figure below shows the angles used for this parameter (a) and some examples of rotations (b). A default value of 90º is used, corresponding to a 3D model without rotation.



Target evaluation height
The height at which flows are analysed is referred to as evaluation height. For CFD simulations at pedestrian level, this evaluation height is usually set between 1.5 and 2m (Blocken, 2015). For drones, for example, this could be set at 10m. A default value of 1.5m is applied.

Unit parameter
One unit in the 3D model is considered as 1 meter. When another unit is required, this can be changed by this parameter.

Domain definition
The minimum domain dimensions outlined by Franke and Baklanov (2007) are taken into consideration. They suggest distances between urban models and domain boundaries as illustrated in the figure below. Note that the height of the tallest building (hmax) is used; however, the largest building dimension (dmax) can also be applied (Liu et al., 2018). For this reason, users are allowed to choose between hmax and dmax.



Furthermore, there is an option to choose between two sets of blockage ratios. The first set describes a blockage ratio BR between the wind-facing area of the 3D city model, and the area of the inlet boundary of the domain (Franke and Baklanov, 2007). BR should remain below 3%. The second set suggests two blockage ratios: the lateral horizontal blockage ratio BRL, and the vertical blockage ratio BRH. BRL is the ratio between the length of the urban area facing the wind direction and the length of the inlet boundary; and BRH is the one between the height of the tallest building and the height of the computational domain. Both BRL and BRH must remain below 17%.

$$BR = \frac{A_{urban}}{A_{domain}}, BR \leq 3\% \tag{G.1}$$

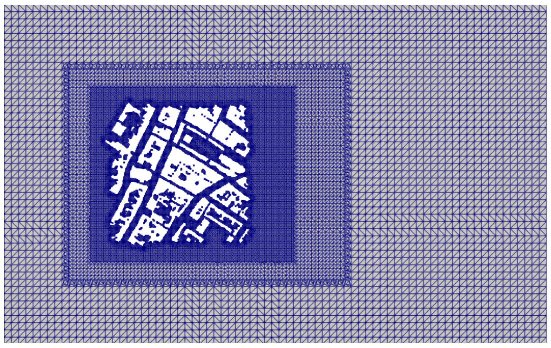
$$BR_L = \frac{L_{building}}{L_{domain}}, BR_L \leq 17\% \tag{G.2}$$

$$BR_H = \frac{H_{building}}{H_{domain}}, BR_H \leq 17\% \tag{G.3}$$

No blockage ratios are selected by default.

Refinement boxes
A refinement box delineates an area where the cell dimensions are reduced to increase the number of cells in which the Navier-Stoke equations can be solved. A minimum number of three refinement boxes is suggested for CFD simulations in urban areas (Franke and Baklanov, 2007). However, in practise, two refinement boxes are often used. For this reason, the option is given to choose between two or three refinement boxes.

In general, with three refinement boxes, the first box is placed directly around the urban area and is a bit higher than the tallest building; the second box surrounds the first and extends slightly toward the outflow boundary; the last box is placed around the second one. With two refinement boxes, only the two first of these boxes are generated. Also, the closer the refinement box is to the urban area, the higher the refinement level of the cells. The figure below shows an example of refinement boxes defined for an urban CFD simulation.



Note that the refinement boxes should maintain the same proportions as the computational domain, except for the one the closest to the urban area.

The dimensions of the refinement boxes can be indicated by setting multiples of hmax or dmax (dependent on which values was used to define the domain) for the inlet, outlet, lateral and top boundaries. For example, the settings box 1: 2 6 2 2 imply 2xhmax (or dmax) between the urban area and the inlet, lateral, and top boundaries of the refinement box, and 6xhmax (or dmax) between the urban area and the outflow boundary of the refinement box.

The following default values are set for each refinement box: box 1: 0.5 0.5 0.5 1.5, box 2: 2 6 2 2, and box 3: 3 9 3 3.

Maximum roughness length
The centre point of the first cell should be placed at a minimum distance of at least one roughness height z0 from the wall (Tominaga et al., 2008), which is the bottom boundary of the computational domain. Therefore, the maximum roughness height used in the CFD simulation must be indicated.

Maximum number of cells
An important limitation is the available memory of the computers (Franke and Baklanov, 2007). Therefore, users have the option to specify the total number of cells in the final mesh. A default value of 30 million cells is set.

Note that the method used approximates this number of cells. There is no guarantee that it matches the actual number generated by OpenFOAM. On average, this number differs by -12.12% when no ground refinement is applied, and -22.20

Maximum cell ratio
The cell ratio is defined as the cell height divided by its width. Ideally, its value should be 1; however, a higher cell ratio is allowed to meet more CFD guidelines, such as ensuring at least 10 cells per cube root of the building volume and building separation (Franke and Baklanov, 2007; Blocken, 2015; Tominaga, 2008). A default value of 1.20 is selected, which corresponds to a cell with a height of 1m and a width of 0.80m.

Maximum building separation and street distribution
A building separation is considered as the distance between two buildings and should measure at least 10 cells (Franke and Baklanov, 2007; Blocken, 2015; Tominaga, 2008). This minimum might lead to excessive mesh complexity as some cities might have narrow streets. Therefore, separations below a specific distance are excluded, which can be set with the maximum building separation parameter. The default value of this threshold is 2m, but could also be user-defined.

To help users determine an appropriate threshold, a histogram displaying the street width distribution can be displayed. This can be achieved by selecting the street distribution option when uploading the 3D model and configuring the parameters.

Target building
A target building is the building on which the focus lies during a CFD simulation.

This parameter is used to define the Region of Interest (RoI), which selects buildings around the target that should be modeled in more detail. This region allows to find a balance between accurate simulation results and computational performance. In fact, the level of detail of surrounding buildings significantly impacts the simulation results around the target building (García-Sánchez, 2021). However, modelling all buildings in the domain would be computationally expensive. The RoI applied is introduced by Liu et al. (2018), which recommend to include buildings within a radius of at least 3·L around the target building.

Users have the option to specify a target building by selecting the target building option and indicating one of the vertices from its ground surfaces. When no target building is specified, its centre is set to the centre of the 3D city model. At the end, the buildings intersecting with the defined RoI are stored.

G.2. Geometric validation parameters

val3dity
Standards were developed to define basic primitives (ISO19107) (ISO, 2019) and provide guidelines for their digitization (OGC, 2016, 2011), with the aim of enhancing the interoperability and exchange of geographical data. As the quality of geometries plays an important role in CFD simulations, verifying them based on the ISO19107 standard could be a "useful

starting point" (Wagner et al., 2015). Therefore, the validation tool named val3dity developed by Ledoux (2018) was implemented.

val3dity is an open-source software that validates 3D primitives based on ISO19107, with the common GIS exception that they need to be linear or planar. More information on this software can be found on the val3dity GitHub page.

val3dity takes into account some user-defined tolerances.

- Snap tolerance: two vertices are considered identical if they are separated by this value or less (default: 0.001),
- Planarity tolerance: ISO19107 requires all vertices of a planar surface to lie on one plane, which is almost impossible with real-world data (Ledoux, 2013). For this reason, val3dity considers this parameter defining the maximum distance between a point and a fitted plane (Ledoux, 2018) (default: 0.001),
- Overlap tolerance: two Solids that overlap or are disconnected by this value or less are considered properly connected (default: -1, which stands for disabled).

Topological relationships thresholds
The validation of topological relationships involves verifying whether building objects are correctly positioned on terrain surfaces.

During this validation, the following buildings are identified:

- Floating buildings (in reality, buildings are always connected to the ground),
- Buildings separated by a specific distance from terrain surfaces.

This distance is defined by the topological relationships threshold, with a default value of 0.5m.

In the absence of terrain features, the ground level is used instead of terrain surfaces. This level corresponds to either a user-defined value or the lowest z-value in the input model. The former can be indicated by selecting the ground level option and inserting the required value.

Sliver triangle threshold
Sliver triangles are triangles with a sliver parameter lower than a specific threshold, which can be indicated with this parameter. This sliver parameter is defined by using the following formula (Artwork Conversion Software, n.d.):

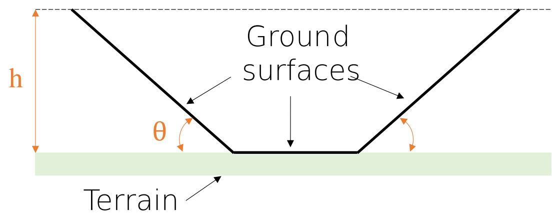
They are identified to help users simplify their geometries, as small geometric details might affect the meshing process.

Short edge threshold
Short edges are edges shorter than a specific threshold (m), which can be indicated with this parameter. They are identified to help users simplify their geometries, as small geometric details might affect the meshing process.

Sharp angle threshold
Sharp angles are angles between two faces smaller than a specified threshold (°), which can be indicated with this parameter. They are identified to help users simplify their geometries, as small geometric details might affect the meshing process.

$$SP = \frac{2 \times area}{perimeter} \tag{G.4}$$

Ground surfaces
For the topological relationships validation, the ground surfaces of buildings are used. Building surfaces lower than a given threshold h (default: 3m) and with an angle smaller than a threshold theta (default: 45°) with regard to the ground are considered as ground surfaces. The figure below illustrates this method.



Note that these values must carefully be selected. In fact, this approach might lead to incorrect results when, for example, buildings contain internal geometries, or have heights lower than h and flat roofs.

G.3. Results

Separate building and terrain validation
Report describing errors and warnings from the separate building and terrain validations performed by the val3dity software. More information can be found on the val3dity GitHub page.

Topological relationships errors
Both OBJ and TXT files are given to users to visualise topological relationships.

- Input model with terrain surfaces:
  - OBJ file showing terrain surfaces and vertices of ground surfaces that are not correctly positioned (i.e. vertices higher than terrain surfaces, or separated with a distance higher than a specific value),
  - TXT file indicating terrain surfaces *t* by their centroids and vertices of ground surfaces by their coordinates that are not correctly positioned. Also, the distance between them is given.

```
n 323149

    vb 90.337000 137.397820 -0.824460
     t 92.088875 138.319479 -0.822822
     d 1.580513
```

- Input model without terrain surfaces:
  - OBJ file showing vertices of ground surfaces that are not correctly positioned with regard to the ground level (i.e. vertices with a z-value higher than ground level, or separated with a distance higher than a specific value). This is done by visualising these vertices and the same vertices but with the z-value of the ground level.
  - TXT file indicating the pairs of vertices previously described with the distance between them.

```
n 16029

     vb 92583.820310 435896.875000 2.715000
     vt 92583.820310 435896.875000 1.000000
      d 1.715000
```

Sharp angles
Both OBJ and TXT files are given to users to visualise sharp angles. In the TXT file, pairs of faces representing these sharp angles are indicated. For each face, their centroid is provided.

```
n 11
f pair
     f1
          v1 92646.15625 436551.96875 15.06800
          v2 92646.15625 436551.96875 17.11500
          v3 92628.45312 436551.28125 17.10000
     f2
          v1 92639.65625 436552.06250 17.11000
          v2 92646.15625 436551.96875 17.11500
          v3 92646.15625 436551.96875 15.06800
     θ    3.05028
```

Sliver triangles
Both OBJ and TXT files are given to users to visualise sliver triangles. In the TXT file, these angles are indicated by their centroids.

```
n 28655
f
      v1 92583.82031 435896.87500 2.71500
      v2 92597.45312 435901.00000 2.71500
      v3 92583.94531 435896.50000 2.71500
      SP 0.19490
```

Short edges
Both OBJ and TXT files are given to users to visualise short edges. In the TXT file, pairs of vertices representing these edges are indicated.

```
n 26270
      v1 92583.94531 435896.50000 2.71500
      v2 92583.82031 435896.87500 2.71500
      l 0.39528
```

Overlapping buildings
Both OBJ and TXT files are given to users to visualise overlapping buildings. In the TXT file, pairs of overlapping buildings are indicated with their corresponding centroids.

```
n 281
      centroid (2D)
      b1 92590.665921 435902.085565
      b2 92610.346579 435893.401981
```

Rotated 3D model
Aligned input model with the incoming flow in OpenFOAM based on the flow direction. This OBJ file can be used to run the CFD simulation.

blockMeshDict
OpenFOAM configuration file in which the background mesh is defined, including domain dimensions and number of cells in each direction.

snappyHexMeshDict
OpenFOAM configuration file in which parameters for the final mesh are defined such as dimensions and refinement levels of the refinement boxes. Additionally, it includes parameters for the ground refinement box when required.

Region of Interest (RoI)
A cylinder representing the RoI and the buildings within this area are provided as two separate OBJ files. The former can be used to recompute the mesh parameters and run the CFD simulation for a better balance between accuracy and computational performance; the latter can be used with the input model to visualise buildings that must be included and the ones that can be excluded. Additionally, a TXT file is returned providing the centroid of buildings within the RoI.

Buildings with insufficient cells
Both a OBJ and TXT file are given to users to show buildings with less than 10 cells per cube root of the building volume. In the TXT file, the location of these buildings are provided by their centroids.

Building separations with insufficient cells
Both a OBJ and TXT file are given to users to show buildings with separations that are shorter than 10 cells. In the TXT file, the location of these buildings are provided by their centroids.

H. blockMeshDict file

This is an example of a *blockMeshDict* file describing a rectangular domain using the coordinates and number of cells defined under *backgroundMesh*.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      blockMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

backgroundMesh
{
    xMin  83705.36188;
    xMax  85684.86188;
    yMin  447014.64375;
    yMax  448266.54375;
    zMin  -2.60500;
    zMax  440.19500;
    xCells  185;
    yCells  117;
    zCells  41;
}

convertToMeters 1;

vertices
(
    ($:backgroundMesh.xMin $:backgroundMesh.yMin $:backgroundMesh.zMin)
    ($:backgroundMesh.xMax $:backgroundMesh.yMin $:backgroundMesh.zMin)
    ($:backgroundMesh.xMax $:backgroundMesh.yMax $:backgroundMesh.zMin)
    ($:backgroundMesh.xMin $:backgroundMesh.yMax $:backgroundMesh.zMin)
    ($:backgroundMesh.xMin $:backgroundMesh.yMin $:backgroundMesh.zMax)
    ($:backgroundMesh.xMax $:backgroundMesh.yMin $:backgroundMesh.zMax)
    ($:backgroundMesh.xMax $:backgroundMesh.yMax $:backgroundMesh.zMax)
    ($:backgroundMesh.xMin $:backgroundMesh.yMax $:backgroundMesh.zMax)
);

blocks
(
    hex (0 1 2 3 4 5 6 7)
    (
        $:backgroundMesh.xCells
        $:backgroundMesh.yCells
        $:backgroundMesh.zCells
    )
    simpleGrading (1 1 1)
);

edges
(
);

boundary
(
    inlet
    {
        type patch;
        faces
        (
            (0 3 7 4)
        );
    }

    outlet
    {
        type patch;
        faces
        (
            (1 5 6 2)
        );
    }

    ground
    {
        type wall;
        faces
        (
            (0 1 2 3)
        );
    }
    top
    {
        type symmetry;
        faces
        (
            (4 7 6 5)
        );
    }

    left
    {
        type symmetry;
        faces
        (
            (0 4 5 1)
        );
    }

    right
    {
        type symmetry;
        faces
        (
            (3 2 6 7)
        );
    }


);

mergePatchPairs
(
);

// ************************************************************************* //
```

I. snappyHexMeshDict file

This is an example of a *snappyHexMeshDict*file with three refinement boxes and ground refinement. When two refinement boxes are applied, the definition of *refinementBox3* can be removed. Also, the refinement levels for the refinement boxes and ground refinement can be decreased by one level. Without ground refinement, the two parts defining *ground* can be removed.

```
/*--------------------------------*- C++ -*----------------------------------*\
  =========                 |
  \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox
   \\    /   O peration     | Website:  https://openfoam.org
    \\  /    A nd           | Version:  7
     \\/     M anipulation  |
\*---------------------------------------------------------------------------*/
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    object      snappyHexMeshDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

#includeEtc "caseDicts/mesh/generation/snappyHexMeshDict.cfg"

castellatedMesh on;
snap            on;
addLayers       off;

geometry
{
    buildings
    {
        type triSurfaceMesh;
        file "Binnenstad.obj";
    }

    ground
    {
        type            searchablePlane;
        planeType       pointAndNormal;
        pointAndNormalDict
        {
            basePoint       (83705.4 447015 -2.605);
            normal          (0 0 1);
        }
    }

    refinementBox1
    {
        type searchableBox;
        min  (84030.09688 447335.64375 -2.60500);
        max  (84639.99687 447945.54375 180.99500);
    }

    refinementBox2
    {
        type searchableBox;
        min  (83922.67288 447228.64375 -2.60500);
        max  (85035.47287 448052.54375 224.19500);
    }

    refinementBox3
    {
        type searchableBox;
        min  (83850.23587 447159.09375 -2.60500);
        max  (85251.93587 448122.09375 720.99500);
    }
};

castellatedMeshControls
{
    features
    (
        { file "Binnenstad.eMesh"; level 1; }
    );

    refinementSurfaces
    {
        buildings
        {
            level (4 4);
            patchInfo { type wall; }
        }

    }

    refinementRegions
    {
        refinementBox1
        {
            mode inside;
            levels ((1E15 3));
        }

        refinementBox2
        {
            mode inside;
            levels ((1E15 2));
        }

        refinementBox3
        {
            mode inside;
            levels ((1E15 1));
        }

        ground
        {
            mode distance;
            levels ((2.02500 4));
        }

    }

    locationInMesh (84030.09688 447335.64375 180.99500);
    nCellsBetweenLevels 4;
    maxGlobalCells 150000000;
}

snapControls
{
    explicitFeatureSnap    true;
    implicitFeatureSnap    false;
}

addLayersControls
{
    layers
    {
        "CAD.*"
        {
            nSurfaceLayers 2;
        }
    }

    relativeSizes       true;
    expansionRatio      1.2;
    finalLayerThickness 0.5;
    minThickness        1e-3;
}

meshQualityControls
{}

writeFlags
(
    // scalarLevels
    // layerSets
    // layerFields
);

mergeTolerance 1e-6;

// ************************************************************************* //
```

Bibliography

About AIJ.

JTS | Documentation.

JTS | FAQ.

Abu-Zidan, Y., Mendis, P., and Gunawardena, T. (2021). Optimising the computational domain size in CFD simulations of tall buildings. *Heliyon*, 7(4):e06723.

AMS. Atmospheric boundary layer - Glossary of Meteorology.

AMS. Macroscale - Glossary of Meteorology.

AMS. Mesoscale - Glossary of Meteorology.

AMS. Microscale - Glossary of Meteorology.

AMS. Troposphere - Glossary of Meteorology.

AMS (n.d.). Glossary of meteorology. `https://glossary.ametsoc.org/wiki/Welcome`.

Arroyo Ohori, K., Ledoux, H., and Peters, R. (2022). *3D modelling of the built environment*, volume v0.8.

Arthur, R. S. and Angevine, W. M. (2023). 5 - What's next: Boundary layer prediction methods. In Hiscox, A. L., editor, *Conceptual Boundary Layer Meteorology*, pages 101–114. Academic Press.

Artwork Conversion Software (n.d.). Sliver Definition and Removal.

Benzi, R. and Toschi, F. (2023). Lectures on turbulence. *Physics Reports*, 1021:1–106.

Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K. H., and Khoo, V. H. S. (2016). The most common geometric and semantic errors in CityGML datasets. IV-2/W1:13–22.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.

Blocken, B. (2014). 50 years of computational wind engineering: Past, present and future. 129:69–102.

Blocken, B. (2015). Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations. 91:219–245.

Blocken, B. and Carmeliet, J. (2006). The influence of the wind-blocking effect by a building on its wind-driven rain exposure. *Journal of Wind Engineering and Industrial Aerodynamics*, 94(2):101–127.

Bogdahn, J. and Coors, V. (2010). Towards an automated healing of 3D urban models.

Brancher, M., Griffiths, K. D., Franco, D., and de Melo Lisboa, H. (2017). A review of odour impact criteria in selected countries around the world. *Chemosphere*, 168:1531–1570.

Britannica, T. E. o. E. (2023). Kinetic energy | Definition, Formula, Units, Examples, & Facts | Britannica.

Christou, M. D. (1998). II.5. - Consequence Analysis and Modelling. In Kirchsteiger, C., Christou, M. D., and Papadakis, G. A., editors, *Industrial Safety Series*, volume 6 of *Risk assessment and management in the context of the seveso II directive*, pages 193–230. Elsevier.

COST (2023). About COST - Funding Research Networking.

CREST Foundation Studies (n.d.). 3. Fluid Dynamics. In *Fundamentals of Fluid Mechanics*.

Ellul, C., Zlatanova, S., Rumor, M., and Laurini, R. (2013). Geometric validation of 3D city models based on standardized quality criteria. In *Urban and Regional Data Management*, pages 203–216. CRC Press, 0 edition.

Franke, J. and Baklanov, A. (2007). *Best Practice Guideline for the CFD Simulation of Flows in the Urban Environment: COST Action 732 Quality Assurance and Improvement of Microscale Meteorological Models*.

García-Sánchez, C., Vitalis, S., Pađen, I., and Stoter, J. (2021). The impact of level of detail in 3d city models for cfd-based wind flow simulations. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLVI-4/W4-2021:67–72.

GEOS. GEOS.

Greenshields, C. J. (2023). User Guide.

Grünbaum, B. (2003). Are Your Polyhedra the Same as My Polyhedra? In Aronov, B., Basu, S., Pach, J., and Sharir, M., editors, *Discrete and Computational Geometry: The Goodman-Pollack Festschrift*, Algorithms and Combinatorics, pages 461–488. Springer, Berlin, Heidelberg.

Habby, J. (n.d.). Scales of motion. https://www.theweatherprediction.com/habyhints3/733/.

Hall, N. (2021). Navier-Stokes Equations.

Hisashi, O., Yasuo, O., and Hitomitsu, K. Wind Load Provisions of the Revised Building Code in Japan.

Hu, H. H. (2012). Chapter 10 - Computational Fluid Dynamics. In Kundu, P. K., Cohen, I. M., and Dowling, D. R., editors, *Fluid Mechanics (Fifth Edition)*, pages 421–472. Academic Press, Boston.

ISO (2019). ISO 19107:2019(en) geographic information — spatial schema.

Karki, S., Thompson, R., and McDougall, K. (2010). Data validation in 3D cadastre. In Neutens, T. and Maeyer, P., editors, *Developments in 3D Geo-Information Sciences*, pages 92–122. Springer Berlin Heidelberg, Berlin, Heidelberg.

Kerr, R. M. (1981). *Theoretical Investigation of a Passive Scalar such as Temperature in Isotropic Turbulence.* PhD thesis. ADS Bibcode: 1981PhDT........88K.

Khawaja, H. and Moatamedi, M. (2018). Semi-implicit method for pressure-linked equations (simple) solution in matlab®. *The International Journal of Multiphysics*, 12:313.

Kováčová, M. and Richtáriková, D. (2020). 8 - Critical forces and collisions. How to solve nonlinear equations and their systems. In Martín-Vaquero, J., Carr, M., Queiruga-Dios, A., and Richtáriková, D., editors, *Calculus for Engineering Students*, Mathematics in Science and Engineering, pages 157–178. Academic Press.

Labetski, A., Vitalis, S., Biljecki, F., Arroyo Ohori, K., and Stoter, J. (2023). 3d building metrics for urban morphology. 37(1):36–67.

Ledoux, H. (2013). On the validation of solids represented with the international standards for geographic information: On the validation of solids represented with the international standards. 28(9):693–706.

Ledoux, H. (2018). val3dity: validation of 3d GIS primitives according to the international standards. 3(1):1.

Liou, W. W. (2008). Chaotic Flows. In Li, D., editor, *Encyclopedia of Microfluidics and Nanofluidics*, pages 246–248. Springer US, Boston, MA.

Liu, S., Pan, W., Zhao, X., Zhang, H., Cheng, X., Long, Z., and Chen, Q. (2018). Influence of surrounding buildings on wind flow around a building predicted by CFD simulations. 140:1–10.

Markatos, N. C. (1986). The mathematical modelling of turbulent flows. *Applied Mathematical Modelling*, 10(3):190–220.

Mirzaei, P. A. and Carmeliet, J. (2013). Dynamical computational fluid dynamics modeling of the stochastic wind for application of urban studies. *Building and Environment*, 70:161–170.

OGC (2011). OpenGIS® implementation standard for geographic information - simple feature access - part 1: Common architecture. OGC 06-103r4, version 1.2.1.

OGC (2016). OpenGIS® geography markup language (GML) encoding standard. OGC 07-036r1, version 3.2.2.

OGC (2021). OGC city geography markup language (CityGML) part 1: Conceptual model standard. OGC 20-010, version 3.0.

OpenFoam. OpenFOAM: User Guide: snappyHexMesh.

OpenFOAM (2010). OpenFOAM guide/Finite volume method (OpenFOAM) - OpenFOAMWiki.

OpenFOAM (2015). OpenFOAM guide/Discretization - OpenFOAMWiki.

OpenFOAM (n.d.). OpenFOAM: API Guide: applications/solvers/incompressible/simpleFoam/simpleFoam.C File Reference.

OpenFoam (n.d.). OpenFOAM: User Guide: OpenFOAM®: Open source CFD : Documentation.

Oxford University Press (n.d.). venturi effect.

Pađen, I., García-Sánchez, C., and Ledoux, H. (2022). Towards automatic reconstruction of 3d city models tailored for urban flow simulations. 8:899332.

Park, G., Kim, C., Lee, M., and Choi, C. (2020). Building Geometry Simplification for Improving Mesh Quality of Numerical Analysis Model. *Applied Sciences*, 10(16):5425.

Piepereit, R., Deininger, M., Kada, M., Pries, M., and Voß, U. (2018). A sweep-plane algorithm for the simplification of 3D building models in the application scenario of wind simulations. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-4/W10:151–156.

Seetha, C. J., Mehta, S. K., Kakkanattu, S. P., Purushotham, P., Betsy, K. B., and Musaid, P. P. (2023). Characteristics of the atmospheric boundary layer during transient conditions of the Indian summer monsoon. *Theoretical and Applied Climatology*.

Shojaei, D., Olfat, H., Quinones Faundez, S. I., Kalantari, M., Rajabifard, A., and Briffa, M. (2017). Geometrical data validation in 3D digital cadastre - A case study for Victoria, Australia. *Land Use Policy*, 68:638–648.

Shukla, A., Singh, A. K., and Singh, P. (2011). A Comparative Study of Finite Volume Method and Finite Difference Method for Convection-Diffusion Problem. *American Journal of Computational and Applied Mathematics*, 1(2):67–73.

SimScale (2023). What Are Navier-Stokes Equations? | SimWiki.

Smith, R. (2014). Sliver Treatment Strategies for CFD | Symscape.

The World Bank (2022). Overview. https://www.worldbank.org/en/topic/urbandevelopment/overview.

Therias, A., Theodoridou, E., Papadimitriou, C., Visser, F., Zhang, F., and Panagiotidou, I. (2022). Removing shared faces in 3d datasets for numerical simulations.

Tominaga, Y., Mochida, A., Yoshie, R., Kataoka, H., Nozu, T., Yoshikawa, M., and Shirasawa, T. (2008). AIJ guidelines for practical applications of CFD to pedestrian wind environment around buildings. 96(10):1749–1761.

Tong, Z., Chen, Y., and Malkawi, A. (2016). Defining the influence region in neighborhood-scale CFD simulations for natural ventilation design. 182:625–633.

UNEP (2021). 5 ways to make buildings climate change resilient. http://www.unep.org/news-and-stories/story/5-ways-make-buildings-climate-change-resilient.

van Oosterom, P., Quak, W., and Tijssen, T. (2005). About Invalid, Valid and Clean Polygons. In *Developments in Spatial Data Handling*, pages 1–16. Springer Berlin Heidelberg, Berlin, Heidelberg.

Wagner, D., Alam, N., Wewetzer, M., Pries, M., and Coors, V. (2015). Methods for geometric data validation of 3D city models. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XL-1/W5:729–735.

Weisstein, E. W. (n.d.). Homeomorphic.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.