Voronoi mesh generation tailored for urban flow simulations

Geomatics Master's Thesis Proposal

Ákos S. Sárkány

Responsible supervisor:	Clara García-Sánchez
Secondary supervisor:	Hugo Ledoux
Date P2:	12.06.2025

June, 2025

1 Introduction

Deriving a 3D virtual representation of the context for urban flow simulations carries two main difficulties. First, the environment has to be captured and reconstructed virtually. Second, the continuous 3D domain has to be discretized for the computations. Researchers at Delft University of Technology developed the application city4cfd (2022) (Pađen et al., 2022, 2024) that automatically reconstructs a Digital Terrain Model and a boundary representation of the buildings in a given area. However, these representations cannot be directly used for CFD without being spatially discretized through meshing.

The discretization should be done by dividing the 3D domain into discrete cells. The shape and structure of a cell in a mesh can vary between simulations — tetrahedral and hexahedral meshes are among the most commonly used, but polyhedral meshes also exist, and they have particular advantages over traditional meshes for certain applications. An important benefit of polyhedral meshes is that they can be generated from a Voronoi diagram constructed over an unorganized set of points. A Voronoi mesh partitions space into cells, where each cell consists of all points closer to a given point in the unorganized structure than to any other point. Given that the unorganized structure of seed points is generated with the correct conditions on a boundary, the constructed Voronoi mesh conforms to that boundary (Merland et al., 2014). This feature is useful for mesh generation, where the goal is to discretize a volume represented by a boundary.

This document proposes an approach to reuse and refine the implementation of existing Voronoi mesh generators to develop one tailored for urban flow simulations. The goal is to generate a polyhedral mesh (to be used with the OpenFOAM[1] open-source CFD software) that conforms to the output geometry of *city4cfd*. The promise of this study is to derive a meshing method that generates a higher (or at least the same) quality mesh faster than traditional mesh methods used for urban flow simulation. The next section reviews the relevant literature. Section 3 introduces the prospective research question and objective. In Section 4, a methodology is proposed to answer the research question. Section 5, lists the necessary tools for the project, and Section 6 presents the schedule for the research.

2 Related research

2.1 Urban flows with OpenFOAM

OpenFOAM is one of the most widely used open-source CFD solvers. Its main advantage is that it offers tools for the entire CFD workflow, including mesh generation and the solution of discretized field equations. The two primary meshing utilities in OpenFOAM are `blockMesh` and `snappyHexMesh`. `blockMesh` creates a structured background mesh, which is then castellated and snapped to the geometry boundaries by `snappyHexMesh` (Gisen, 2014). Snapping is an iterative process that adjusts mesh vertices to better align with the input surface. While it aims to conform closely to the boundaries, it may not always fully capture small-scale features, particularly when the local cell size is larger than the surface detail.

[1]OpenFOAM: https://www.openfoam.com/

Blocken (2015) provides general guidelines for CFD simulations in urban environments, including recommendations for domain size. Based on the study, the minimum domain dimensions should be

$$(x + 20h) \times (y + 10h) \times 6h$$

where $x \times y$ is the dimensions of the interest region and h is the height of the tallest building. Since urban CFD typically requires large domains and high mesh resolution near buildings, the resulting meshes can contain a substantial number of cells.

2.2 Automatic reconstruction of urban environments with city4cfd

Pađen et al. (2022, 2024) introduce city4cfd, 2022, a tool for automatic reconstruction of urban environments. It uses point cloud data or existing geometry to derive a boundary representation for the context of the CFD simulation. The output includes terrain and building surfaces within a defined area of interest, enclosed by a bounding region. Surface patch types are imprinted onto the terrain, and different surface categories can be exported as separate files. However, this representation is not directly suitable for CFD. To address this, the authors use `blockMesh` and `snappyHexMesh` to generate a discretized mesh based on the city4cfd output. Alternatively, an unstructured mesh could be constructed using Delaunay triangulation or Voronoi tessellation.

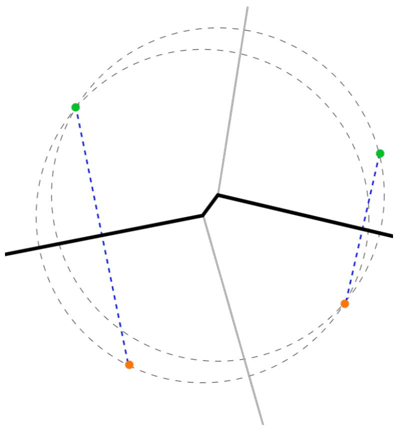
2.3 Meshing with Delaunay Triangulation/Tetrahedralization

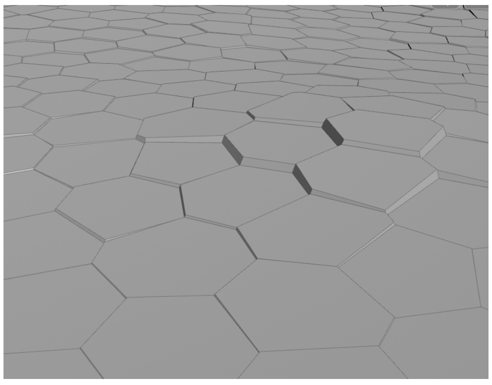
Delaunay triangulation-based mesh generation has been researched for some time now. Baker (1989) introduced a methodology to derive a conforming tetrahedral mesh given the triangulated boundary of an object. Their approach directly uses the constrained Delaunay tetrahedralization for the mesh. Ju (2007) improves this approach by using centroidal-Voronoi Delaunay triangulation. This way, they achieve more uniform cells. Later, Wang et al. (2015) introduces a fast variant of this triangulation method. Triangulation and tetrahedralization are important concepts, however, they yield significantly more cells than polyhedral meshing, which can cause longer simulation runtime.

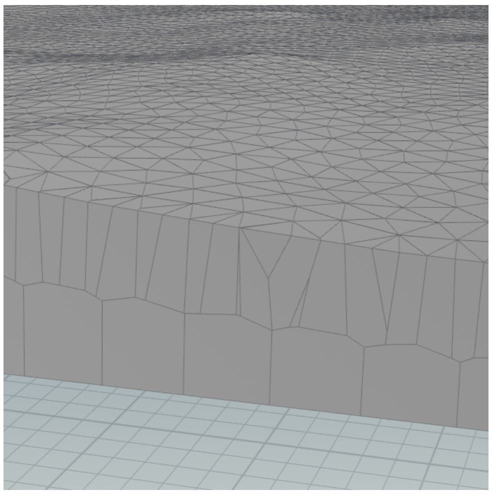
2.4 Meshing with Voronoi Tesselation

Ebeida and Mitchell (2012) discusses an approach to partition 3D domains into Voronoi cells. They use random points scattered over the volume, and to ensure conformity, they clip the Voronoi cells by the boundary. This is not optimal since the clipping requires extra computation and can yield non-convex cells.

Abdelkader et al. (2018, 2020) introduce VoroCrust, an algorithm for generating a Voronoi mesh without clipping, and prove that their approach conforms to 3D surface geometry. Their method focuses on optimal sampling conditions for objects with arbitrarily sized and shaped surface features, while also preserving sharp angles. Seed points are recovered using the weighted Delaunay triangulation of the surface mesh, with each vertex assigned a sphere whose radius reflects the local geometric feature size.


(a) When the 4 seed points are not equidistant from a single point, but two, a small facet forms between unpaired seed points (see (b)). Adapted from Abdelkader et al. (2020).


(b) When the naive mirroring approach is applied to the terrain, it yields undesired results.


(c) Terrain generated with the VoroCrust approach. The algorithm used here is a simpler version that does not prune covered and half-covered seeds. This approach yields a triangular boundary. The main issue with this approach is illustrated in (d).

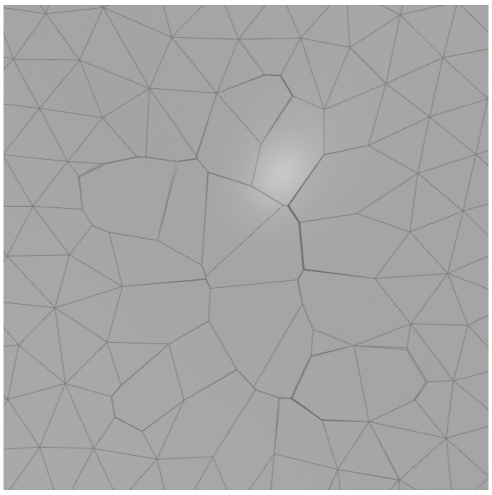

(d) If the sphere triplet is covered (or half covered) by a fourth sphere, the resulting boundary surface will be incorrect.

Figure 1: Issues with the naive mirroring approach on curved boundaries and meshing with the VoroCrust approach.

The union of balls centered at a triangle's vertices produces overlapping regions, and in certain configurations, the boundary of this union contains exactly two intersection points where all three balls meet. If a fourth sphere does not cover this triple overlap, a pair of seed points is generated at the intersection. This ensures that all seed points associated with a common vertex are equidistant from that vertex. Since this condition holds for each vertex, the problematic configuration illustrated in Figure 1a cannot occur. Furthermore, they explain why a naive mirroring approach fails on curved surfaces. Figures 1a and 1b illustrate the issue with this method.

LaForce et al. (2023) evaluate the performance of the VoroCrust algorithm for generating meshes in subsurface simulations. Their results show that VoroCrust outperforms (flexed-)hexahedral meshing methods, particularly in execution speed, where it consistently performs better.

Berge et al. (2019) explore using Voronoi meshes (referred to in their work as PEBI grids)for modeling subsurface well sites and fracture planes. Their approach is similar to VoroCrust since they also use the intersection of vertex-spheres, emphasizing the importance of using simplices for the boundary surface. This allows vertex sphere radii to be chosen semi-randomly, with the only requirement being that the three vertex spheres must overlap for each triangle. For polygons with more than 3 vertices, the sphere radii have more constraints since they can overlap multiple points. The radii must be chosen precisely to avoid the potential creation of degenerate seeds.

3 Research Objective

Research Question:

What are the necessary steps to generate a boundary conforming 3D Voronoi mesh for urban CFD, from the output of city4cfd?

To answer the main research question, first, the following *research sub-questions (rsq.)* must be answered:

1. How to generate boundary conforming Voronoi tessellation in 3D?
    a. Which method is the most efficient for the terrain?
    b. Which method is the most efficient for the buildings?
    c. How to preserve boundaries of surface patches during mesh generation?
2. Is it possible to use structured generator points in certain mesh regions? How can the structured and unstructured areas be connected?
3. Is the resulting mesh of good quality to be used for urban CFD? Can boundary conditions be set up appropriately on the resulting mesh?

3.1 Scope of the research

This research focuses on deriving the 3D mesh from the existing boundary representation of the context. This representation is assumed to be valid, with no self-intersections or holes. Repairing or adjusting to invalid input geometries is not in the scope of this research; such geometries might produce unexpected, potentially wrong results. Furthermore, running CFD simulations might not be achieved within the given time frame of the project. The primary goal is to derive a representation of the mesh containing all the geometric and boundary-type (at least the input file name) information necessary for CFD, but which still requires semantic translations before it can be used in numerical solvers, like OpenFOAM. This can be either a set of seeds (aka. generator points) which partition the domain into Voronoi cells, or a set of polygons defining the boundaries of each cell.

3.2 Why cannot VoroCrust, 2023 be used as is?

The procedure for generating seed points should use the geometric output of city4cfd, 2022 as a boundary constraint. Preliminary experiments of this study showed that not every output of city4cfd can be meshed by the VoroCrust, 2023 software. The program did not terminate within a reasonable time on certain inputs and was likely stuck in an infinite loop.

Another limitation of the VoroCrust, 2023 software is its inability to achieve a gradual (average) cell size transition close to boundaries. There is a possibility to constrain max cell size, but the preferred outcome is to have larger cells in open areas and smaller cells only close to the boundaries.

Further constraints are posed by only taking a single input file. This way, it cannot set up boundary types (to later be used for boundary conditions in CFD) during the meshing process. However, post-processing can address this by associating boundary cells with their nearest surface polygons.

4 Methodology

This section outlines the procedure for generating seed points of an unstructured Voronoi mesh, based on the geometric output of city4cfd (2022). The input consists of separate surface geometry files, each corresponding to a specific patch type. For seed generation, the input is separated into 3 groups with different characteristics: cured boundaries (terrain), planar boundaries (buildings), and the bounding volume (bounding-box).

The following subsection addresses curved boundaries and introduces a methodology to answer *rsq. 1.a* and *1.c.* Subsection 4.2 focuses on planar surfaces, which constitute building geometry, and outlines techniques related to *rsq. 1.b.* Subsection 4.3 then presents methodologies targeting *rsq. 2.* Finally, Subsection 4.4 discusses evaluation metrics for assessing the results *(rsq. 3).*

4.1 Seed generation on curved boundaries

Terrain is generally a curved surface with few sharp edges. To accurately capture this geometry, the mesh should conform to the smooth surface represented by its discretized boundary. Triangular surface meshes offer the most flexibility in following surface curvature, as three points are always coplanar, whereas polygons with more vertices impose additional coplanarity constraints. Furthermore, using 2D simplices on the boundary of a 2-manifold allows looser conditions on vertex-sphere radii (Berge et al., 2019). Therefore, the boundary of the Voronoi mesh should be a surface triangulation of the terrain.

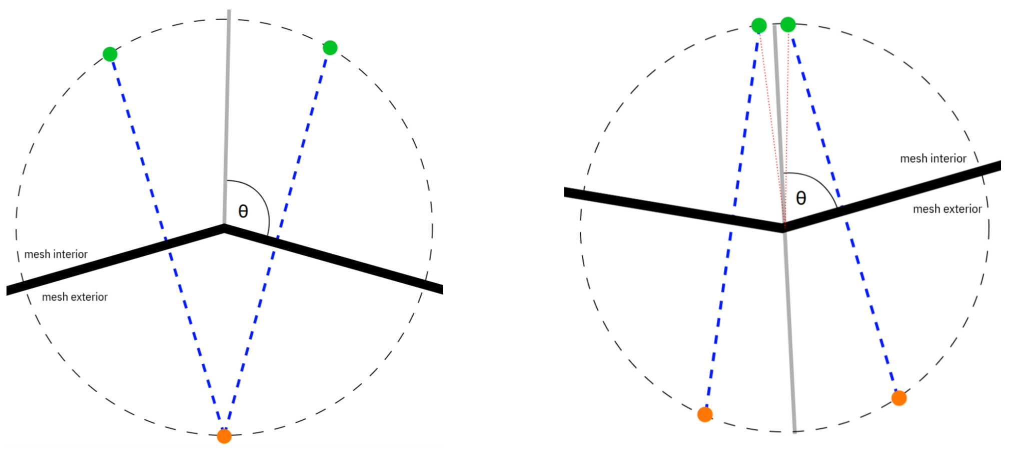
The terrain may consist of multiple surface types (aka. patches). The boundaries between the patches must be preserved since every patch type has its own characteristic. A constrained Delaunay triangulation of the terrain surface should be computed, using the surface patch boundaries as additional constraints. Then, the sampling method of the VoroCrust algorithm (Abdelkader et al., 2020) can be applied to generate Voronoi cells whose boundaries conform to this triangulation. The algorithm will be used to produce seeds for surfaces only, not to fill the interior of the mesh.

4.2 Seed generation on planar boundaries

Unlike terrains, buildings are typically composed of planar surfaces, with curved features being rare. When curves do exist, city4cfd discretizes them into polygonal segments. In other words, each non-planar edge of a building is considered to be a sharp edge. Therefore, seeds can be generated directly on the surface of each polygon without requiring an offset from the plane, since the surface has zero curvature.

Two methodologies will be tested for generating seed points for building surfaces. The first follows the approach used for the terrain, utilizing the VoroCrust algorithm. This method is known to produce valid results as long as the input is a solid (Abdelkader et al., 2020; LaForce et al., 2023). It will be applied to generate seeds for the building surfaces.

The second is a novel approach based on mirroring across planar surfaces. Since all non-planar edges are assumed to be sharp, the boundary of a building can be



(a) Concave corner ($\theta > 90°$):
In this case, it is possible to *collapse* the two exterior seeds into one. Thus, no extra facet can form (Figure 1a) since 3 points determine a unique circumcenter. The exterior cell is removed before the simulation. It preserves surface boundaries between the cells inside the mesh.

(b) Convex corner ($\theta < 90°$):
Seeds inside the mesh might be collapsed into one at a corner if the surface edge is not a boundary between surface patches. If the surface edge is on a boundary, the generator seeds must be placed equidistant from the normal axis (grey).

Figure 2: Alternative method to preserve sharp features. A green dot represents a generator seed inside the mesh, and an orange dot represents a generator on the outside. Outside seeds (and cells) are for generation only; they should be removed before the CFD simulation.

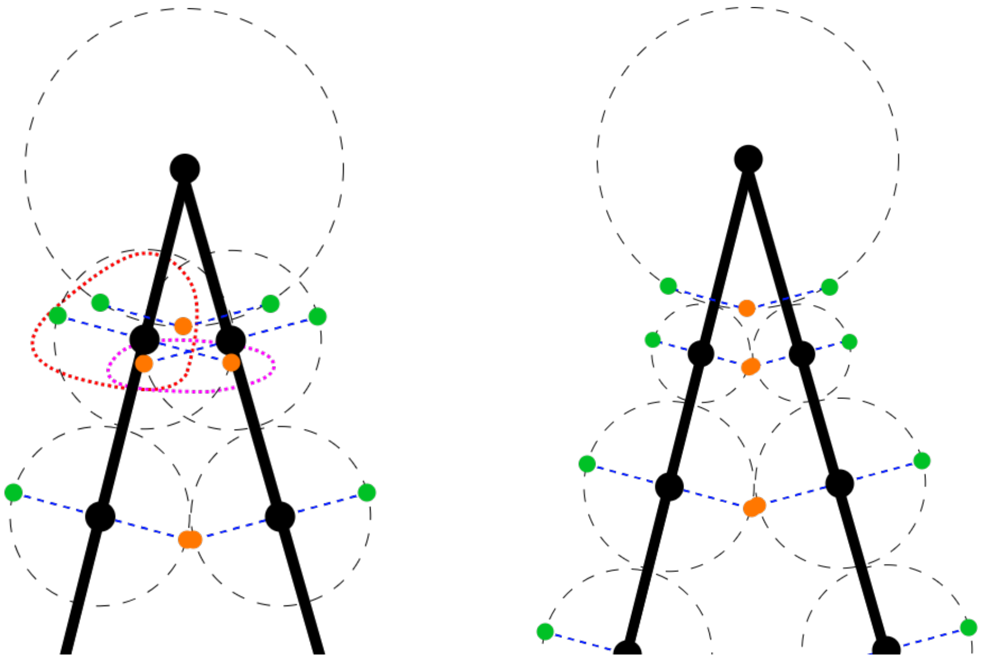


Figure 3: It is crucial to have the correct offsetting distance, especially near sharp corners. Otherwise, the offsetting can move the seeds too close or across the boundary (purple). The surface seeds must have a minimal distance from a corner, otherwise, seeds might pair incorrectly (red). This happens when seed points are inside the spheres. The figure on the right illustrates a setup with correct sampling.

segmented into n planar polygons. Seed points are randomly scattered along the polygons, and a *conforming centroidal-Voronoi Delaunay triangulation (CfCVDT)* (Ju, 2007) is computed for each planar facet. This ensures the points are distributed uniformly along the surface.

The points on each polygon are then duplicated and shifted by the same distance (per polygon) along the normal of the planar surface, with shifting occurring along parallel lines. This way, non-paired seeds cannot get closer to each other, ensuring that unintended seed pairings are avoided on the surface of a polygon. However, close to the polygon boundary, additional care has to be taken when placing the seeds. Therefore, surface seeds "too close" (based on sampling density and *local feature size* (Abdelkader et al., 2018)) to the boundary must be eliminated (Figure 3).

A corner in three dimensions occurs when $n \geq 2$ polygons intersect. For $n = 2$, the result is an edge (a line segment); for $n > 2$, the intersection defines a point. Figure 2 illustrates how sharp features can be preserved. The illustration can be interpreted as a side profile of the $n = 2$ case in 3D, although the concept generalizes to any number of faces n. For a given n, a concave corner requires $n + 1$ generators. In such cases, the collapsed seeds produce a single cell on the exterior of the mesh, which is removed before running CFD simulations (2a).

If the corner is convex and lies on the boundary between surface types, the interior seeds *cannot* be collapsed into a single seed, as doing so would fail to preserve the boundary. This scenario requires a pair of seeds for each of the n faces, resulting in a total of $2n$ seeds (2b). If the corner is not on a surface boundary, the seeds may collapse into one, again yielding $n + 1$ generators.

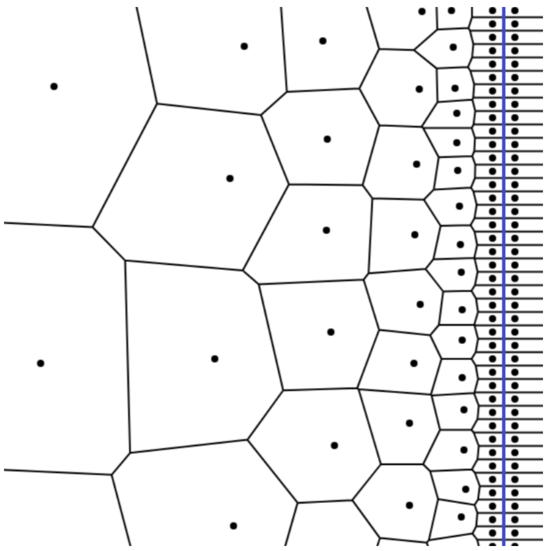


Figure 4: Gradual cell size transition achieved by generating seeds more sparsely further from the boundary in 2D. A similar method can be applied in 3D. Voronoi diagram generated with https://cfbrasz.github.io/Voronoi.html

4.3 Seed generation inside the mesh

VoroCrust generates polyhedral cells by placing random seeds throughout the volume at a specified density. While this approach produces consistent polyhedral cells, a potentially faster alternative is to restrict polyhedral cells to regions near boundaries and use hexahedral cells in open areas. A hexahedral grid could be sufficient in those regions, since the more complex flow happens in the proximity of buildings.

The domain is divided into uniform cells, yielding a grid of hexahedral cells. Cells lying on or near the boundary are identified ("near" is defined based on the mesh generation parameters) and used to generate Voronoi regions for improved boundary conformity. The rest remain as standard hexahedra.

The density (or *fineness*) of the mesh should gradually increase closer to the boundaries. This can be achieved in two ways:

1. Split the hexahedra closest to (but not on) the boundary into smaller cells, thus achieving a gradual transition. Then generate Voronoi seeds with high density only in one layer away from the boundary.

2. Keep the sizes of hexahedra fixed. Generate seeds more densely closer to the boundary (see Figure 4).

To evaluate the optimal strategy, both methodologies will be implemented and compared. Furthermore, a version using only unstructured points will be implemented to serve as a base case during the evaluation of the voxelization.

4.4 Mesh quality assessment

The effectiveness of the algorithm can be evaluated based on two key factors: execution time and the quality of simulations performed on the resulting mesh. Execution time is straightforward to assess by measuring and comparing it against established meshing tools such as `SnappyHexMesh`; this will serve as the primary metric.

Evaluating simulation accuracy is more challenging, as running full numerical simulations is likely beyond the scope of this project. Sadrehaghighi (2023) outlines several methods for assessing mesh quality. In the context of this research, the most important criterion is how closely the generated mesh follows the boundary geometry, which can be evaluated visually or through algorithmic analysis. This will also be compared against the meshes generated by `SnappyHexMesh`. For the comparison, meshes generated with a similar level of detail will be used.

5 Third-party tools and resources

To ensure fast runtime, the meshing program should be implemented in C++. CGAL (https://www.cgal.org/) will be used for geometric computations, which efficiently implement spatial queries and algebraic computations. The source code of the VoroCrust, 2023 algorithm might be (partially) reused during the implementation to enable meshing of terrains. Given that not all steps from the VoroCrust algorithm are planned to be used, it might be beneficial to include or implement only the necessary elements of the algorithm. The city4cfd, 2022 software will be used to generate input files for testing.

6 Schedule

From	Until	Task
		P1: Planning
Apr-18	May-05	Literature Review: Voronoi mesh generation
May-05	May-22	Experiments with existing mesher: VoroCrust
May-12	May-27	Planning & Scheduling
May-22	Jun-05	Writing Graduation Plan
	Jun-12	*P2: Project Kick-off*
		Summer break
Sep-01	Sep-19	Implement surface-seed sampling: VoroCrust approach for terrain & buildings
Sep-01	Sep-19	Implement interior-seed sampling variants
Sep-19	Oct-01	Initial merge of seeds & experiments
Oct-01	Oct-20	Implement surface-seed sampling: Novel approach for buildings
		P3: Midterm meeting
Oct-20	Oct-31	Finalize merge and fix issues
Nov-03	Nov-14	Evaluate results
Sep-01	Nov-21	Finish final draft of Thesis

		P4: Final evaluation
Dec-01	Dec-12	Final evaluation
Dec-01	Dec-19	Finalize Thesis
Dec-15	Jan-09	Prepare P5 Presentation
		P5: Public presentation
Jan-12	Jan-23	Public presentation

Table 1: The preliminary schedule of the project.

References

Abdelkader, A., Bajaj, C. L., Ebeida, M. S., Mahmoud, A. H., Mitchell, S. A., Owens, J. D., & Rushdi, A. A. (2018). Sampling conditions for conforming voronoi meshing by the vorocrust algorithm. *LIPIcs: Leibniz international proceedings in informatics*, *99*, 10.4230/LIPIcs. SoCG. 2018.1. https://doi.org/10.4230/LIPIcs.SoCG.2018.1

Abdelkader, A., Bajaj, C. L., Ebeida, M. S., Mahmoud, A. H., Mitchell, S. A., Owens, J. D., & Rushdi, A. A. (2020). Vorocrust: Voronoi meshing without clipping. *ACM Transactions on Graphics (TOG)*, *39*(3), 1–16. https://doi.org/10.48550/arXiv.1902.08767

Baker, T. J. (1989). Automatic mesh generation for complex three-dimensional regions using a constrained delaunay triangulation. *Engineering with Computers*, *5*, 161–175.

Berge, R. L., Klemetsdal, Ø. S., & Lie, K.-A. (2019). Unstructured voronoi grids conforming to lower dimensional objects. *Computational Geosciences*, *23*, 169–188. https://doi.org/10.1007/s10596-018-9790-0

Blocken, B. (2015). Computational fluid dynamics for urban physics: Importance, scales, possibilities, limitations and ten tips and tricks towards accurate and reliable simulations [Fifty Year Anniversary for Building and Environment]. *Building and Environment*, *91*, 219–245. https://doi.org/https://doi.org/10.1016/j.buildenv.2015.02.015

city4cfd. (2022). City4cfd: A cityjson-based workflow for CFD simulations by the 3D Geoinformation Group at Delft University of Technology [Accessed: 2025-05-07]. https://github.com/tudelft3d/City4CFD

Ebeida, M. S., & Mitchell, S. A. (2012). Uniform random voronoi meshes. *Proceedings of the 20th international meshing roundtable*, 273–290. https://doi.org/10.1007/978-3-642-24734-7

Gisen, D. (2014). Generation of a 3d mesh using snappyhexmesh featuring anisotropic refinement and near-wall layers. *ICHE 2014. Proceedings of the 11th international conference on hydroscience & engineering*, 983–990.

Ju, L. (2007). Conforming centroidal voronoi delaunay triangulation for quality mesh generation. *Inter. J. Numer. Anal. Model*, *4*, 531–547.

LaForce, T., Ebeida, M., Jordan, S., Miller, T. A., Stauffer, P. H., Park, H., Leone, R., & Hammond, G. (2023). Voronoi meshing to accurately capture geological structure in subsurface simulations. *Mathematical Geosciences*, *55*(2), 129–161. https://doi.org/10.1007/s11004-022-10025-x

Merland, R., Caumon, G., Lévy, B., & Collon-Drouaillet, P. (2014). Voronoi grids conforming to 3d structural features. *Computational Geosciences*, *18*(3), 373–383. https://doi.org/10.1007/s10596-014-9408-0

Pađen, I., García-Sánchez, C., & Ledoux, H. (2022). Towards automatic reconstruction of 3d city models tailored for urban flow simulations. *Frontiers in Built Environment*, *8*. https://doi.org/10.3389/fbuil.2022.899332

Pađen, I., Peters, R., García-Sánchez, C., & Ledoux, H. (2024). Automatic high-detailed building reconstruction workflow for urban microscale simulations. *Building and Environment*, *265*, 111978. https://doi.org/https://doi.org/10.1016/j.buildenv.2024.111978

Sadrehaghighi, I. (2023). *Mesh assessment & quality issues* (tech. rep.). CFD Open Series. https://doi.org/10.13140/RG.2.2.25026.94404/3

VoroCrust. (2023). Vorocrust-meshing [Accessed: 2025-04-12]. https://github.com/sandialabs/vorocrust-meshing

Wang, B., Khoo, B. C., Xie, Z., & Tan, Z. (2015). Fast centroidal voronoi delaunay triangulation for unstructured mesh generation. *Journal of computational and applied mathematics*, *280*, 158–173. https://doi.org/10.1016/j.cam.2014.11.035