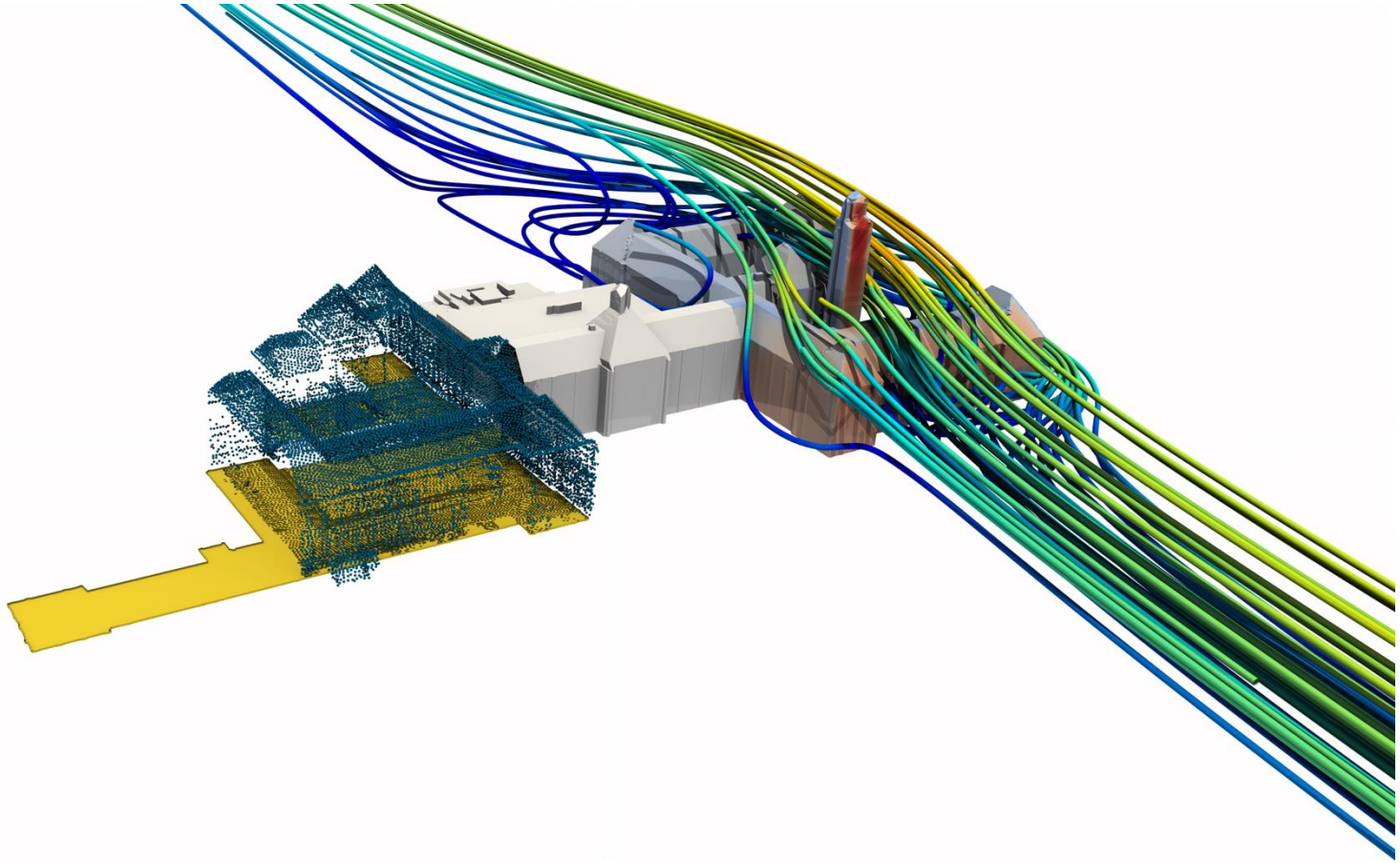


City4CFD:
Automated 3D city reconstruction for different scales in urban flow simulations

Clara García-Sánchez, Akshay Patil, Ivan Pađen, Hugo Ledoux
Delft University of Technology

3D geoinformation
Department of Urbanism
Faculty of Architecture and the Built Environment
Delft University of Technology

Who is here today?

Clara García-Sánchez

Akshay Patil

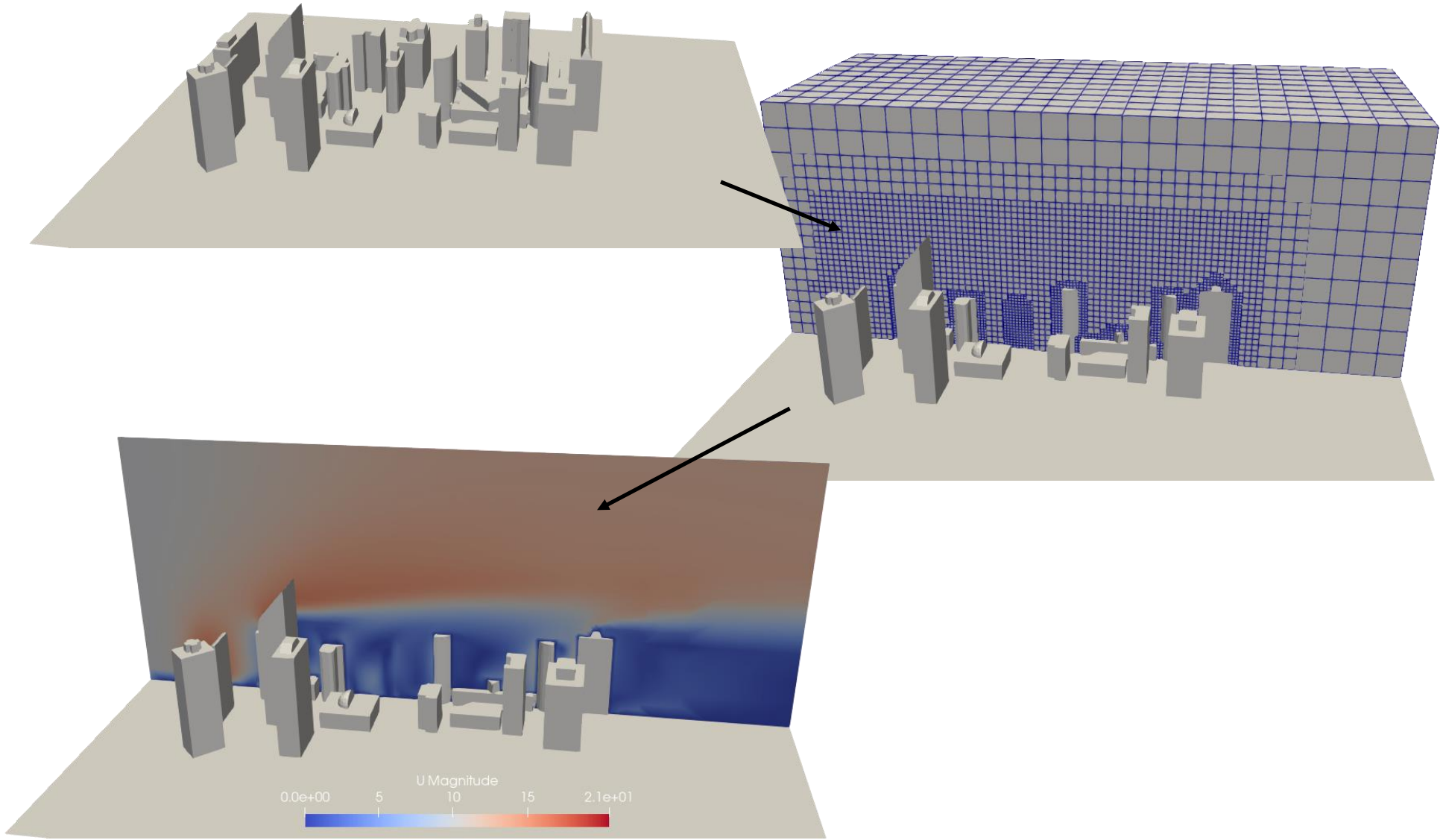
Ivan Pađen
(Lead Author)

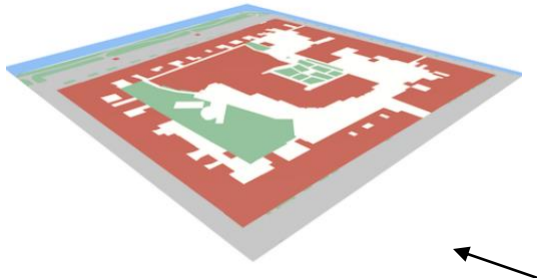
Hugo Ledoux

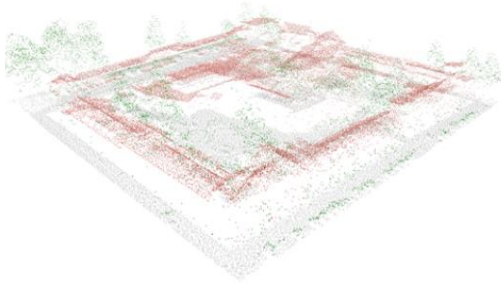
City4CFD

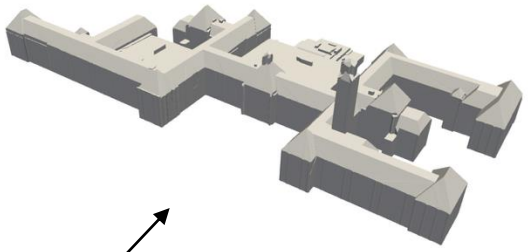
Urban flow geometries

What is specific to urban flows









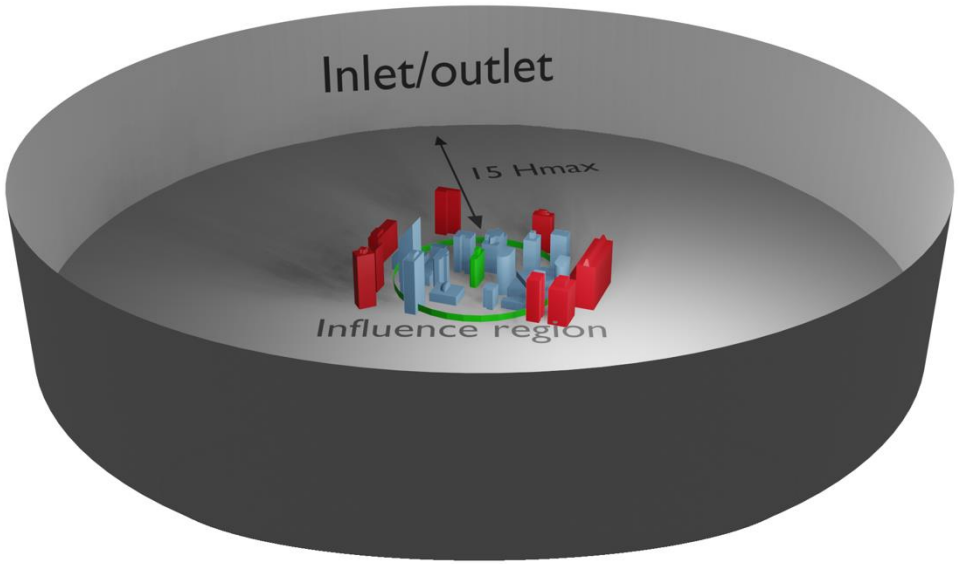
2D topo

point cloud
or
elevation data

existing
geometries

minimum user effort

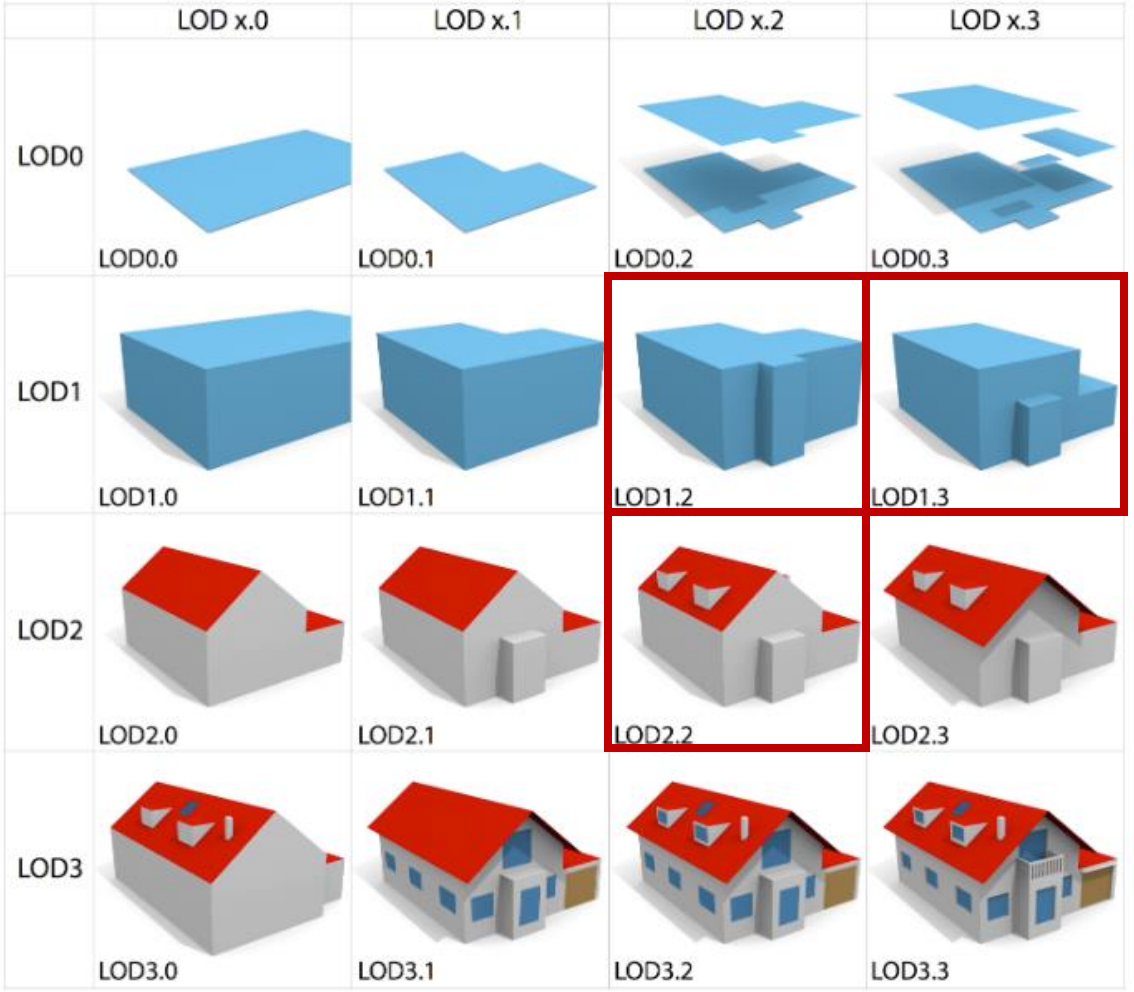
no mesh-breaking errors



- Building of interest/influence region
- Reconstructed buildings
- Virtual/parametrised buildings

Levels of detail

"TU Delft LoD's" City4CFD

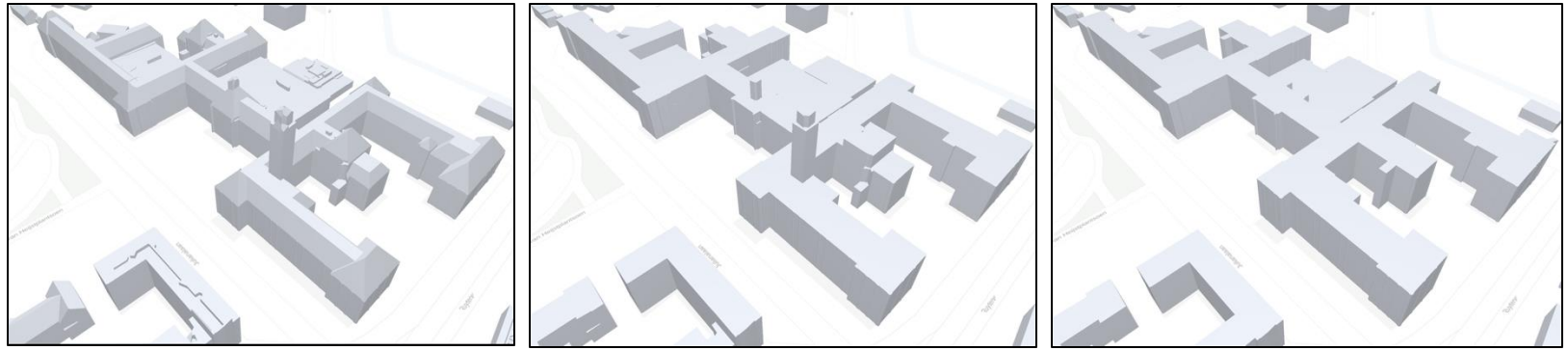


Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. Computers, Environment and Urban Systems, 59, 25-37. https://doi.org/10.1016/j.compenvurbsys.2016.04.005

High-detailed reconstruction

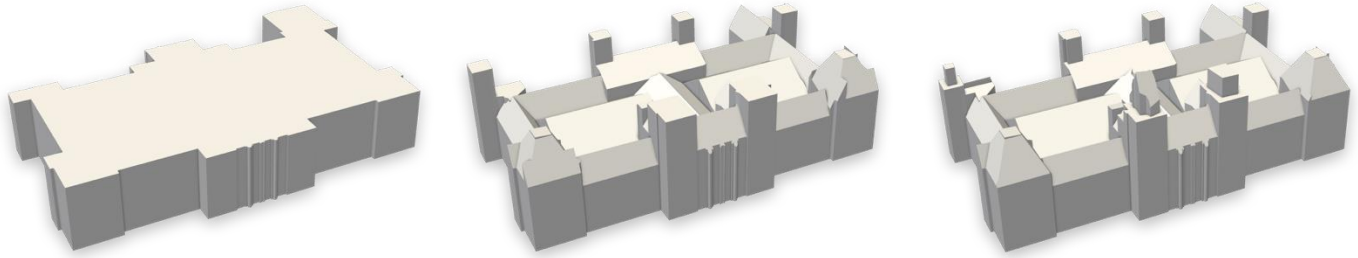
Multiple LoDs

Increasing Geometric Detail

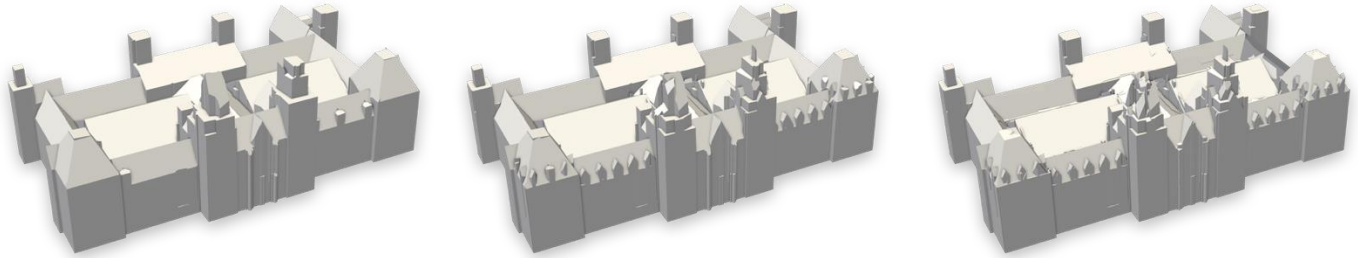


LoD 2.2 LoD 1.3 LoD 1.2

Complexity through graph-cut optimization



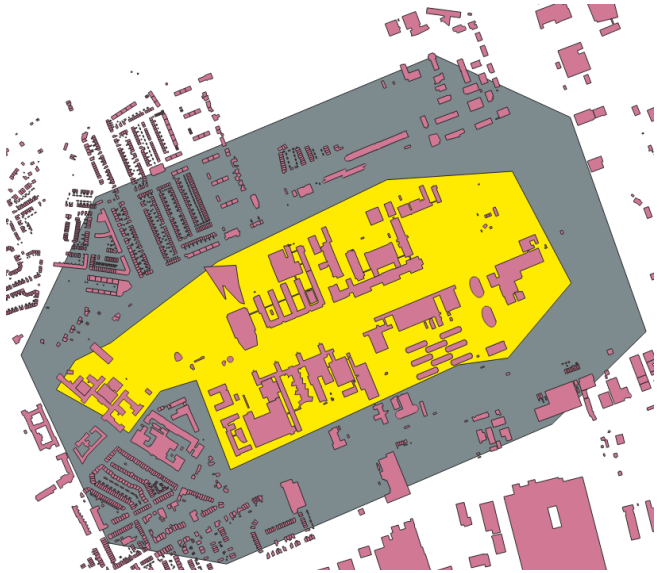
$\lambda = 0.01$ $\lambda = 0.2$ $\lambda = 0.4$



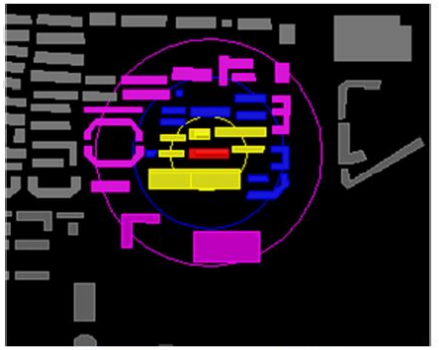
$\lambda = 0.6$ $\lambda = 0.8$ $\lambda = 0.99$

What City4CFD does – boundaries & region of interest

Manual

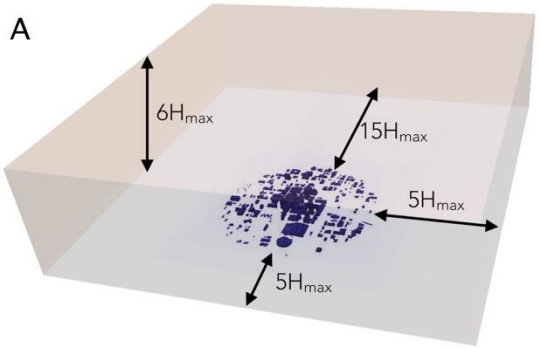


Region of interest BPG



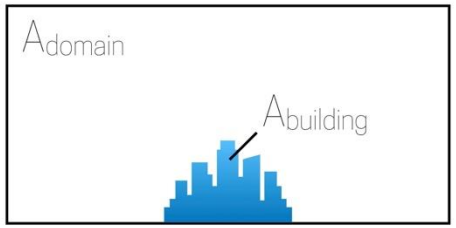
Domain BPG

A



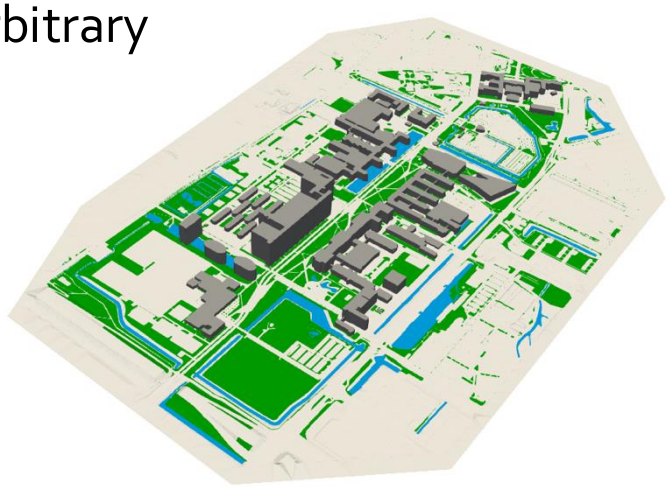
B

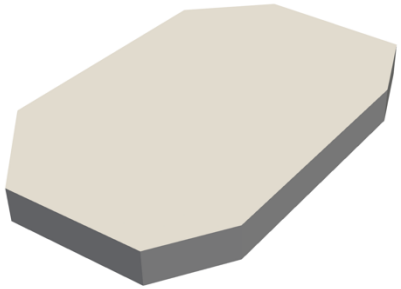
$$BR = \frac{A_{building}}{A_{domain}} < 3\%$$



Domain types

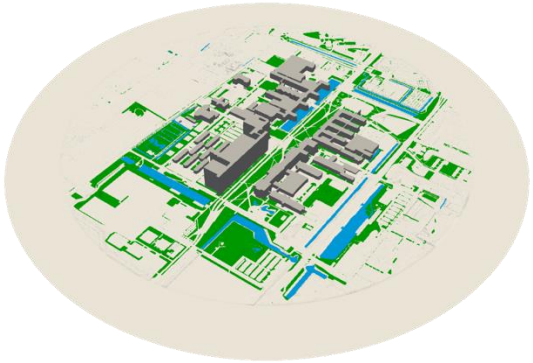
Arbitrary



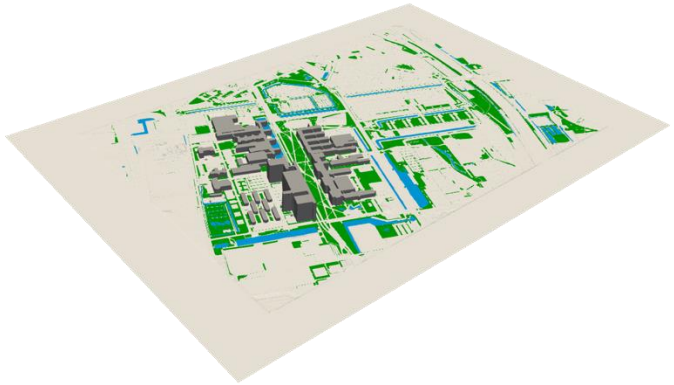


BPG

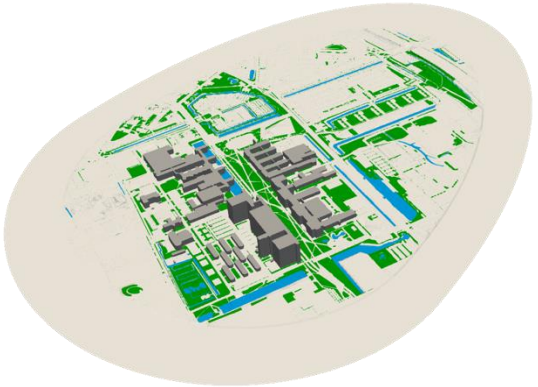
Round



Rectangular*



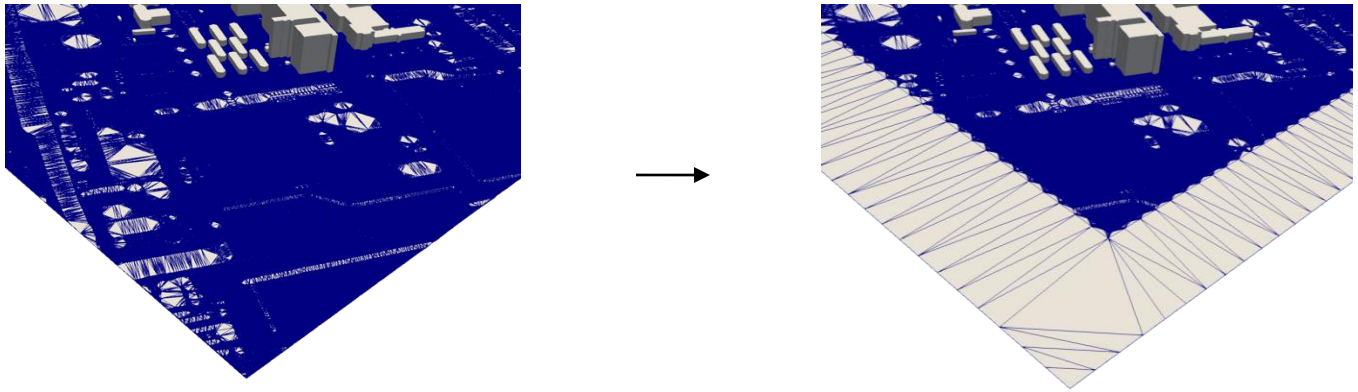
Oval*



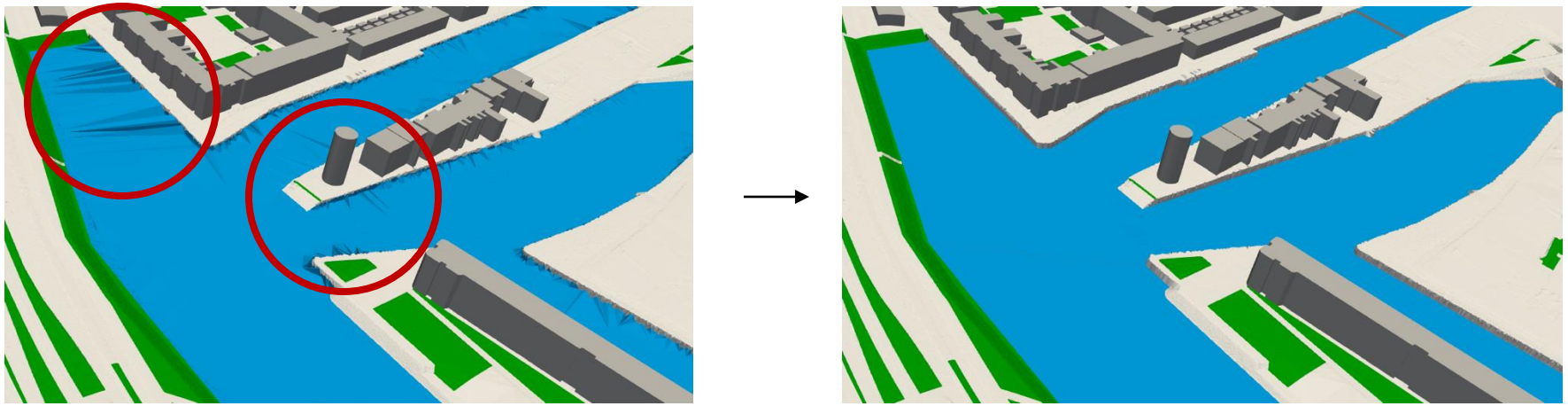
*Adjusted to prescribed flow direction

Other features

Domain edge buffer

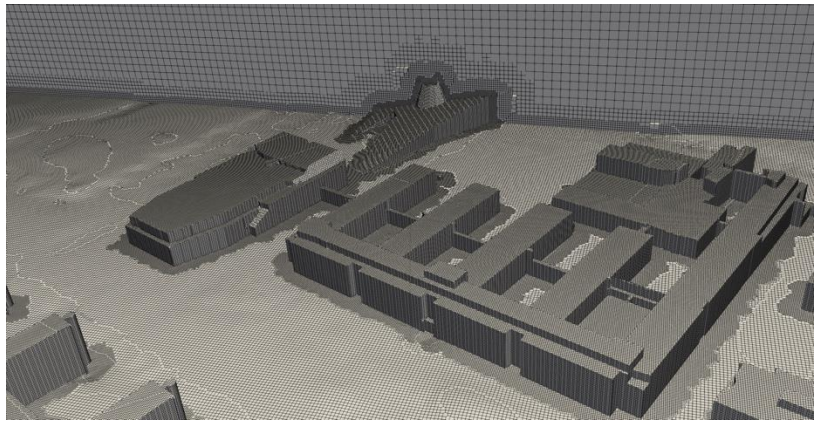


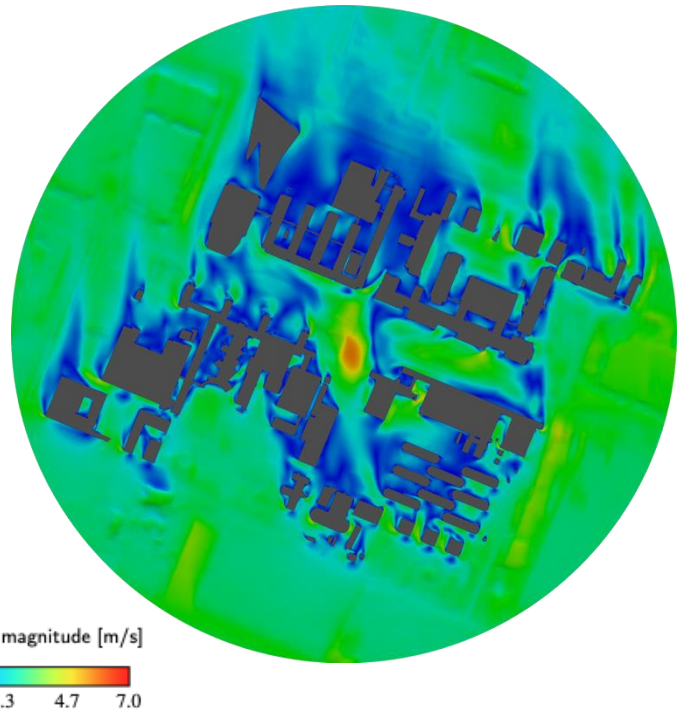
Surface layer flattening – useful for water



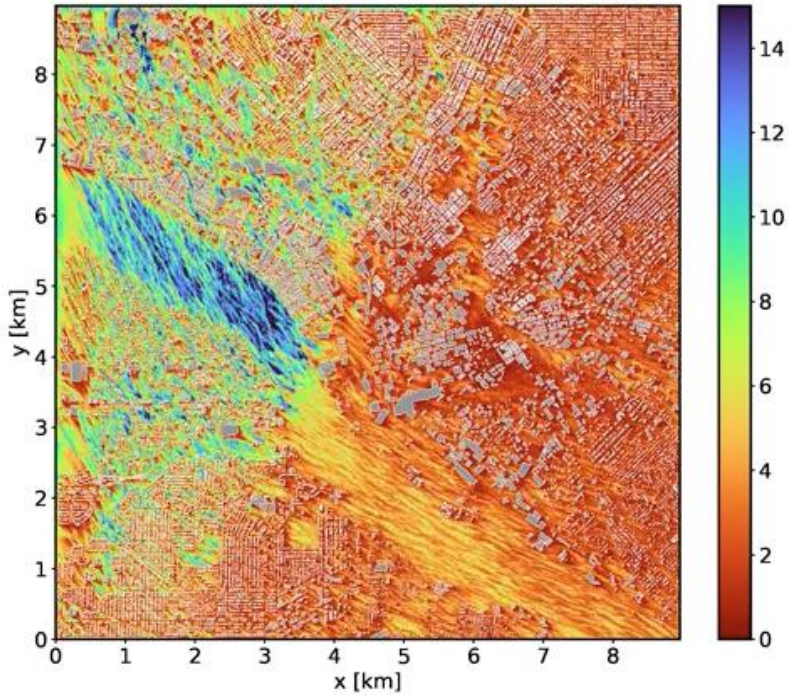
City4CFD in practice

TU Delft (1 km x 1 km)





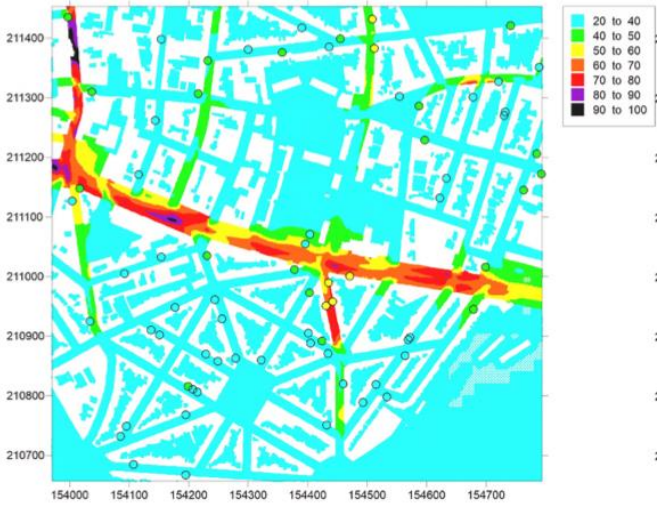
Downtown Dallas, TX (8 km x 8 km)



City4CFD by others

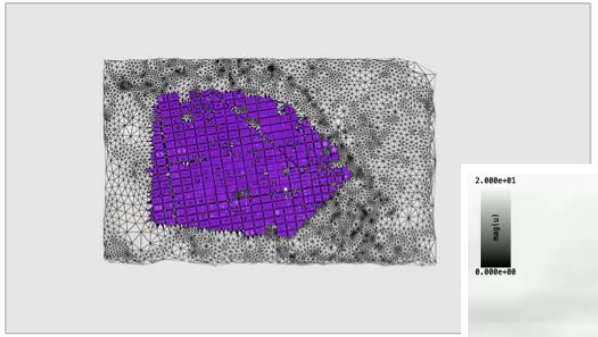
Antwerp

F. Martin et al. (2024)
https://doi.org/10.1016/j.scitotenv.2024.171761



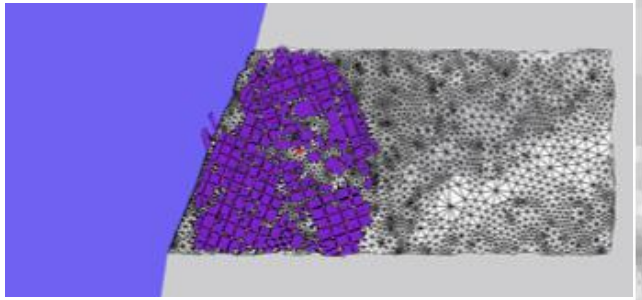
San Francisco, CA

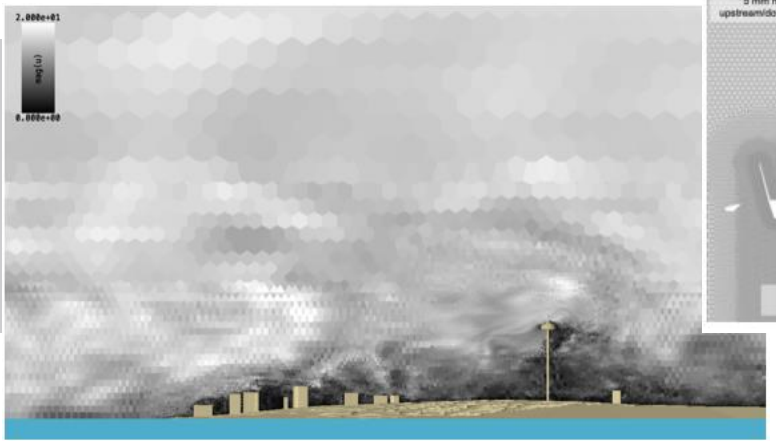
J. Hochschild, PhD thesis
https://purl.stanford.edu/sm262cb0838

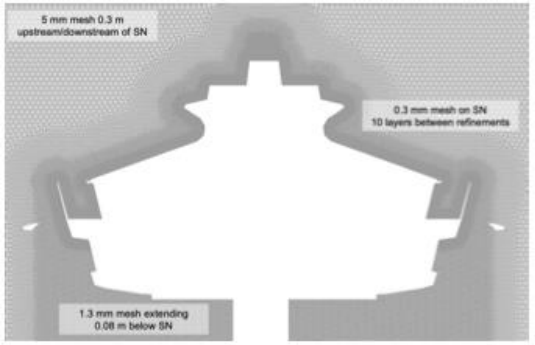


Seattle, WA

J. Hochschild, PhD thesis
https://purl.stanford.edu/sm262cb0838







Any questions up to now?

Let's do some simple pre-checks:

- WSL?
- Paraview?
- QGIS?
- Python?

Download the required git repositories:

- City4CFD repository:

  https://github.com/tudelft3d/City4CFD

- Workshop repository:

  https://github.com/gsclara/ICUC_Workshop

Follow the steps in the repository:

macOS

Mac users can install City4CFD through Homebrew:

```
brew install tudelft3d/software/city4cfd
```

Docker

We offer built Docker images for every release, available at the Docker Hub. Running the docker script for the first time will pull the docker image from the Docker Hub repository.

ONLY MAC USERS

Follow the steps in the repository:

Dependencies are generally available in Linux distributions, e.g. in Debian/Ubuntu/Mint:

```
sudo apt-get install libmpfr-dev libgmp-dev libboost-all-dev libeigen3-dev libomp-dev libgdal-
```

ONLY LINUX/WSL USERS

CGAL can be directly downloaded from the release page (-library). No install is necessary, only the path to the unzipped folder is required (see below).

In macOS you can install all dependencies with Homebrew:

```
brew install cmake boost cgal eigen libomp gdal
```

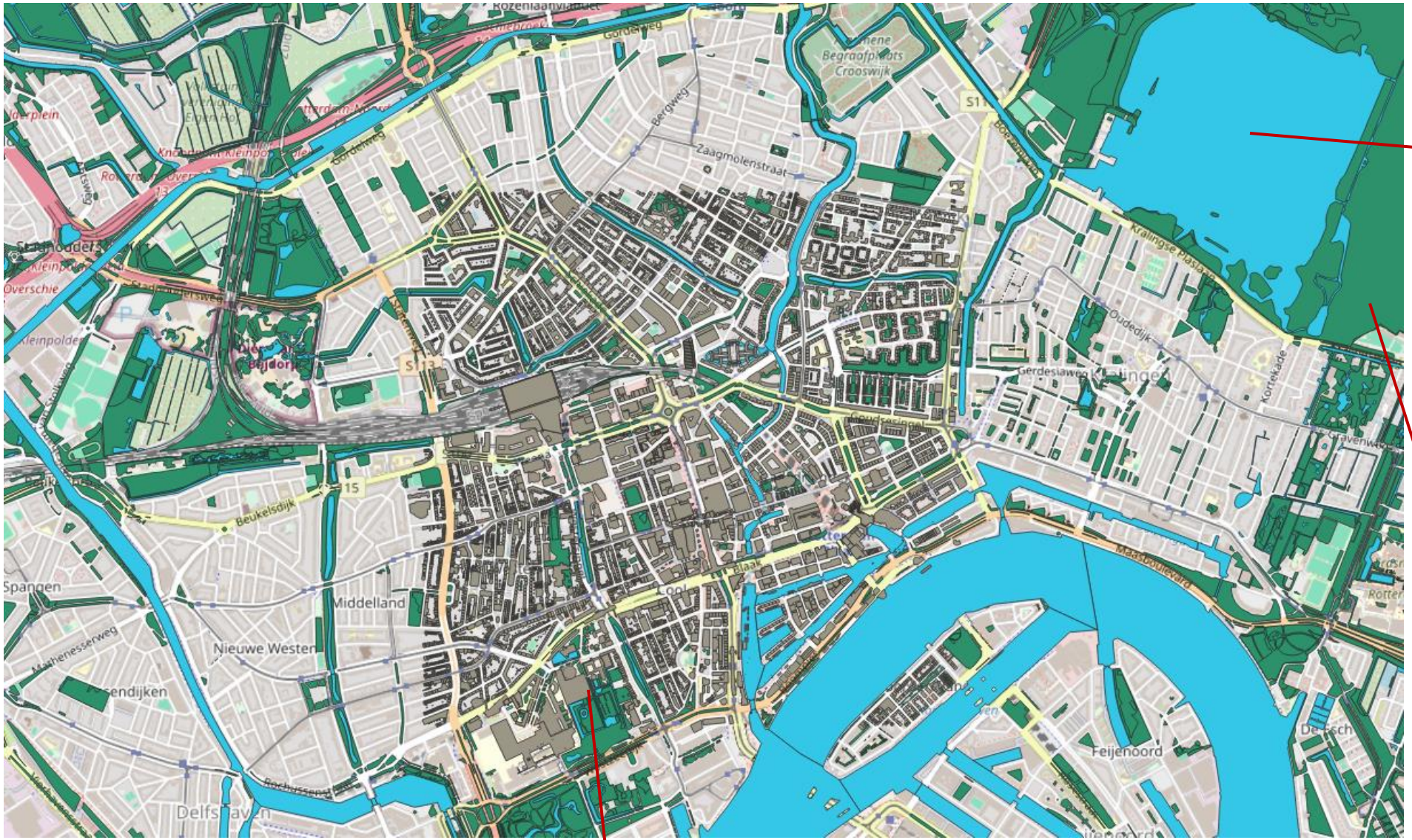
ONLY MAC USERS

To build City4CFD, do the following:

```
mkdir build && cd build
cmake .. -DCGAL_DIR=/path/to/cgal/dir
make
./city4cfd
```

ALL USERS

Reconstructing Rotterdam

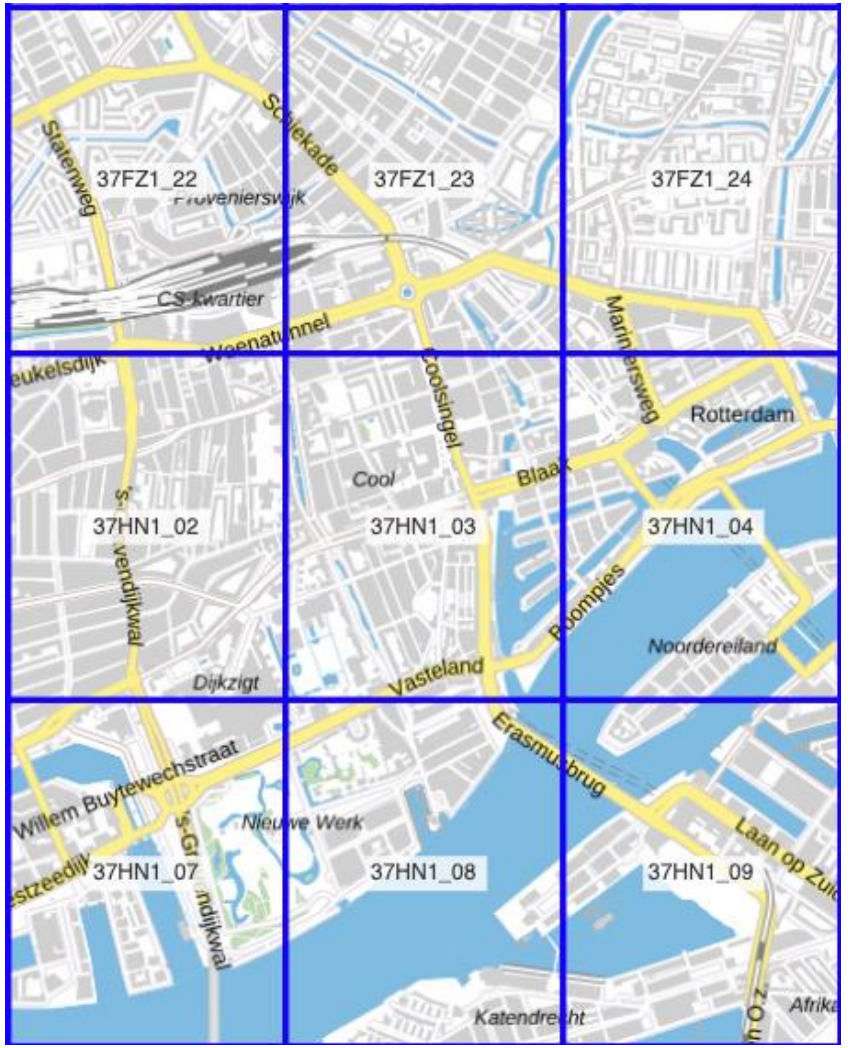


Reconstructing Rotterdam – point clouds

- Geotiles:

  https://geotiles.citg.tudelft.nl/

- Download all the tiles in the image



Reconstructing Rotterdam –the code

tudelft3d / City4CFD

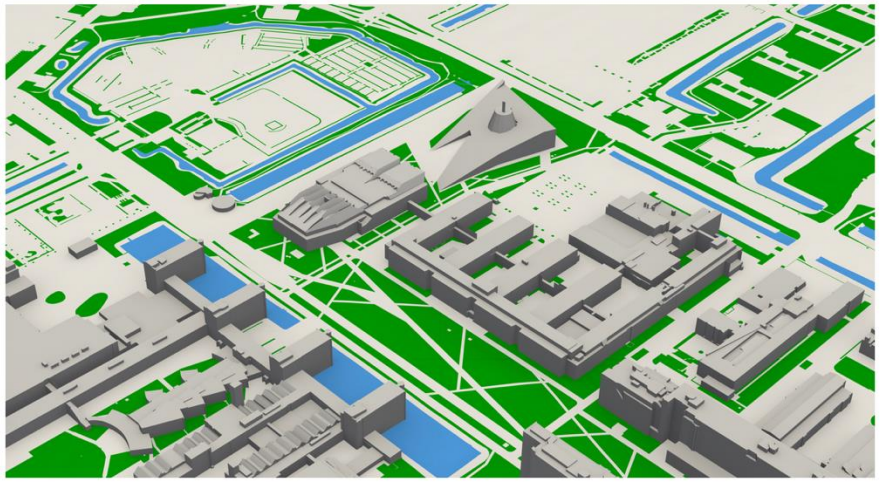
Type / to search

Code | Issues 5 | Pull requests 1 | Discussions | Actions | Projects | Wiki | Security | Insights | Settings

Home

Edit | New page

Akshay Patil edited this page on Apr 24 · 15 revisions



Generating geometries for microscale computational wind engineering simulations is a tedious task for academic and application purposes alike. Researchers and engineers often have to rely on existing 3D geometries which were not made with computational fluid dynamics in mind, leading to a mainly manual and time-consuming task of geometry preparation.

Our aim is to address this issue with a purpose-built software, which we called City4CFD. It combines different input data typical in GIS, such as point clouds and 2D polygons, to create a 3D geometry, while keeping the requirements of CFD packages in mind.

In this documentation, you can find information on how the reconstruction is handled, as well as the information on the preparation of input data and the walkthrough of reconstruction parameters.

Pages 11

Clone this wiki locally

`https://github.com/tudelft3d/City`

Reconstructing Rotterdam – config file

Configuration File

Ivan Pađen edited this page on Jun 3, 2024 · 21 revisions

The configuration file is in JSON format. It is a necessary input and it is validated against a schema.

A few examples of configuration files are in the demo folder.

This section of the documentation provides a walkthrough of the configuration file and available options for the reconstruction. An example of the configuration file containing all options is as follows:

```
//-- CITY4CFD CONFIGURATION FILE --//
{
  "point_clouds":
  {
    "ground": "point_cloud/sampled_ground_1m.ply",
    "buildings" : "point_cloud/sampled_buildings_1m.ply"
  },
  "polygons" : [
    {
      "type": "Building",
      "path": "polygons/tudcampus.geojson",
      "unique_id": "gid"                      // Optional - use building ID from the polygon file
      "height_attribute": "height",           // Optional - use height from the polygon file for reconstructi
      "floor_attribute":  "num_floors",       // Optional - use number of floors for reconstruction
      "floor_height": 3,                      // Floor height, required if 'floor_attribute' is defined
      "height_attribute_advantage": false,    // In case height attribute takes precedence over other reconst
      "avoid_bad_polys": false,               // Optional - ignore or try to reconstruct problematic (non-sim
      "refine": false                         // Optional - refine surface
    },
    {
      "type": "SurfaceLayer",
      "path": "polygons/Vegetation.geojson",
      "layer_name" : "Vegetation"             // Optional - choose a layer name
    },
```

https://github.com/tudelft3d/City4CFD/wiki/Configuration-File

Follow the steps in the repository:

Getting started

The folder *examples* contains example datasets you can run for your first reconstruction. You can run your first reconstruction from the `/examples/TUD_Campus` folder by typing:

```
mkdir results
../../build/city4cfd config_bpg.json --output_dir results
```

in case of building from a source.

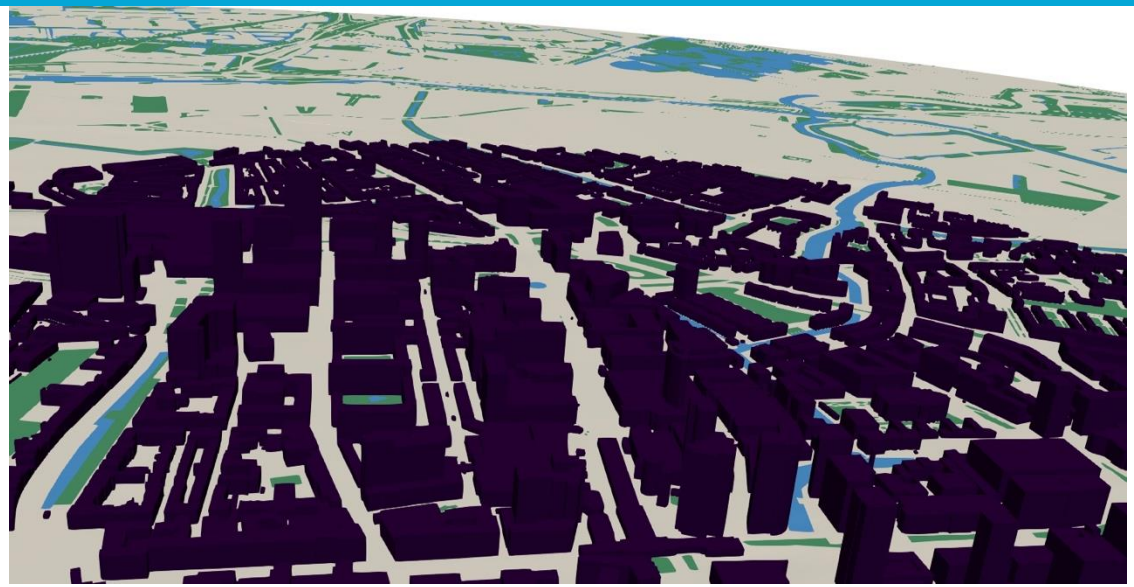
To run through a Docker container, you can use one of the scripts in `docker/run/`. The script with the extension `.sh` can be used in Linux and macOS, the one with the extension `.ps1` in Windows Powershell, and the last one with `.bat` in Windows Command Prompt. You have to run a script (you can copy it beforehand) from the root directory of the project (e.g. `examples/TUD_Campus`), and the arguments are the same as for the compiled executable, e.g.:

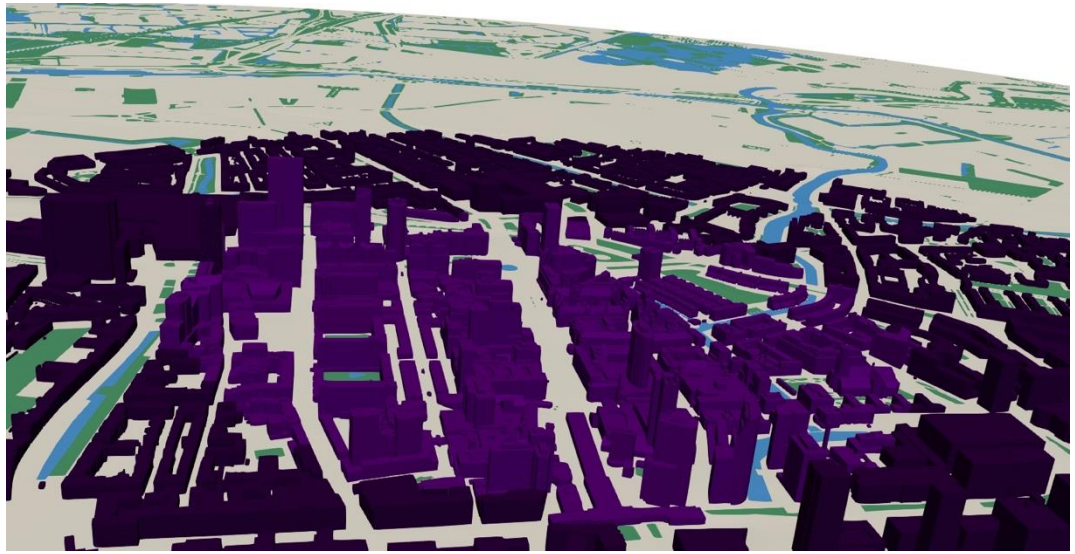
```
../../docker/run/city4cfd_run.sh city4cfd config_bpg.json --output_dir results
```

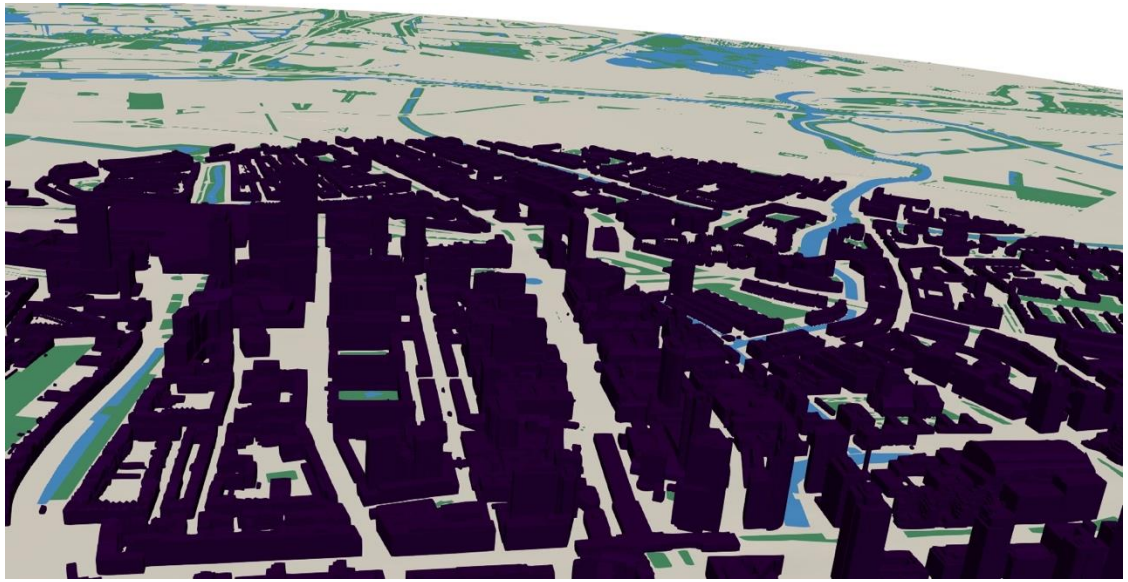
The script pulls the `latest` release from the Docker Hub. For a specific release, replace `latest` in the script with the released version tag, e.g. `0.1.0`. In Linux systems, you will probably have to run the command as a sudo unless you create a 'docker' group and add users to it.

More information on the project can be found in the documentation.

Reconstructing Rotterdam – final result

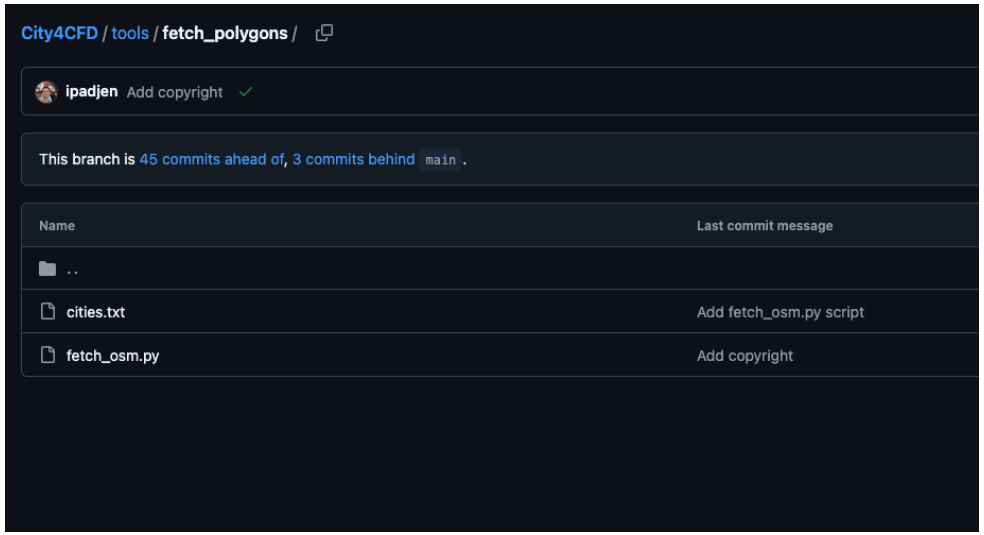




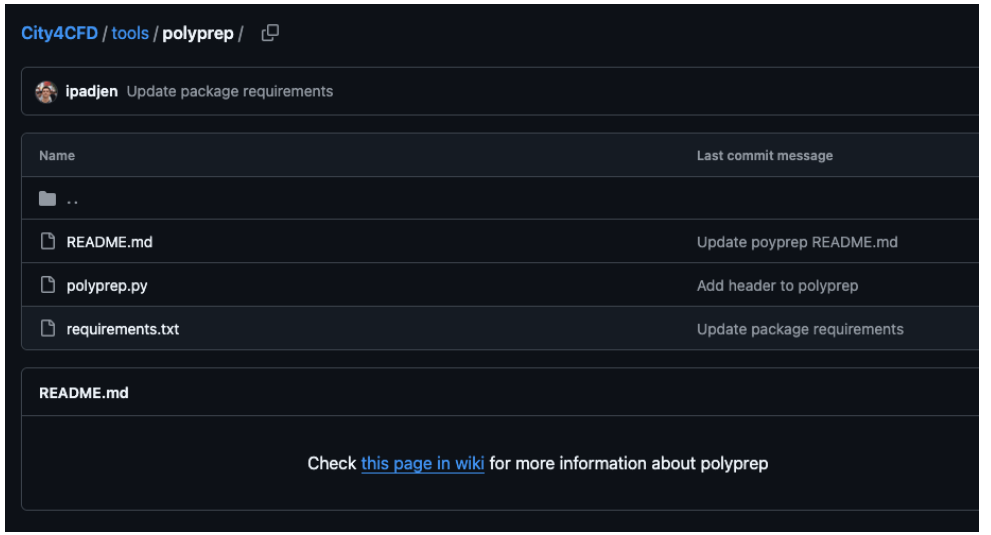


Some Tips/Tricks and Showcase

- Point Cloud availability - Global North ☑ Global South, not as data-rich
  - OSM building height information
  - Local municipality data registry
  - Digital Terrain Model/ Digital Elevation Models (Raster -2.5D)
- Polygon retrieval and Simplification
  - fetch_osm.py – Pulls polygons (Buildings, Vegetation, Water, Forest etc.) from OSM data
  - polyprep.py – Simplifies polygons based on user input



Currently in `develop` branch



Available under `tools`

Using fetch_osm.py

cities.txt input file

City4CFD / tools / fetch_polygons / **cities.txt**

AkshayPatil1994 Add fetch_osm.py script ✓

Code | Blame | 3 lines (3 loc) · 163 Bytes

```
1   Chicago, USA, 41.8755616, -87.6244212, EPSG:6455
2   Barcelona, Spain, 41.39563705715212, 2.161987803576256, EPSG:25831
3   Denver, US,39.7392364, -104.984862, EPSG:4269
```

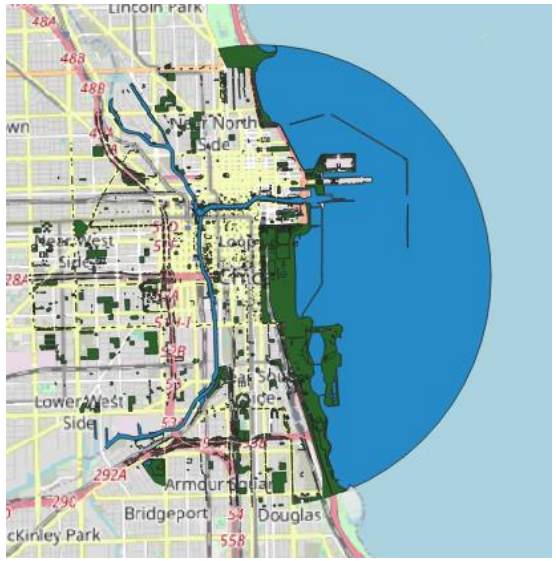
Name of the city

2/3 Letter Country Code

<center-latitude>, <center - longitude> In EPSG: 4326

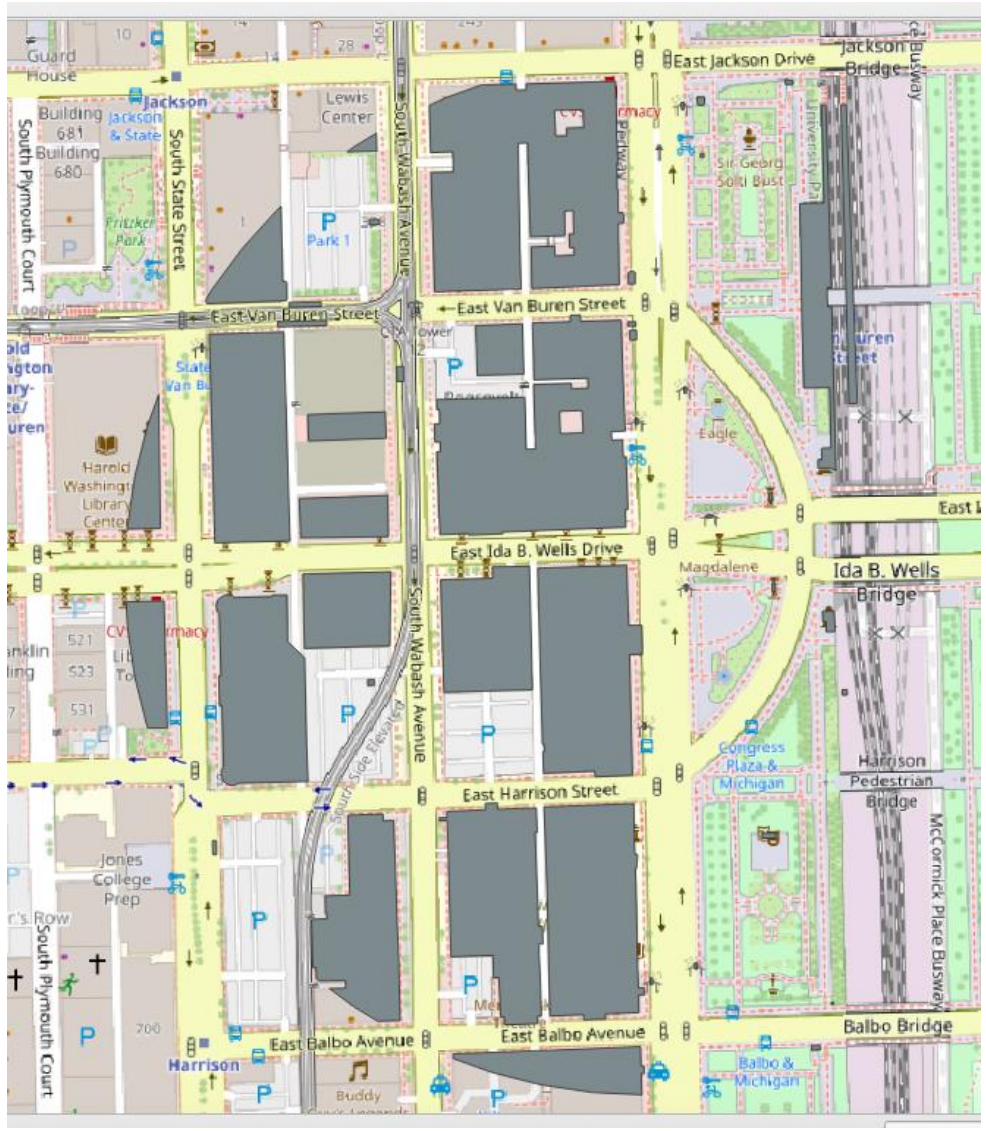
EPSG code Must be same as point cloud

```python
#
# Importing the required libraries
import osmnx as ox
import numpy as np
import os
import geopandas as gpd
from shapely.geometry import Point
from geopy.geocoders import Nominatim
from pyproj import CRS
#
# USER INPUT PARAMETERS
#
Hmax = 230                  # Maximum height of the building the region of interest
rbuildings = 1200           # Radius for building polygons
rpolygons = 4000            # Radius for non-building polygons (only active when Hmax < 0)
outdir = "data"             # Define the output directory where polygons are downloaded
```

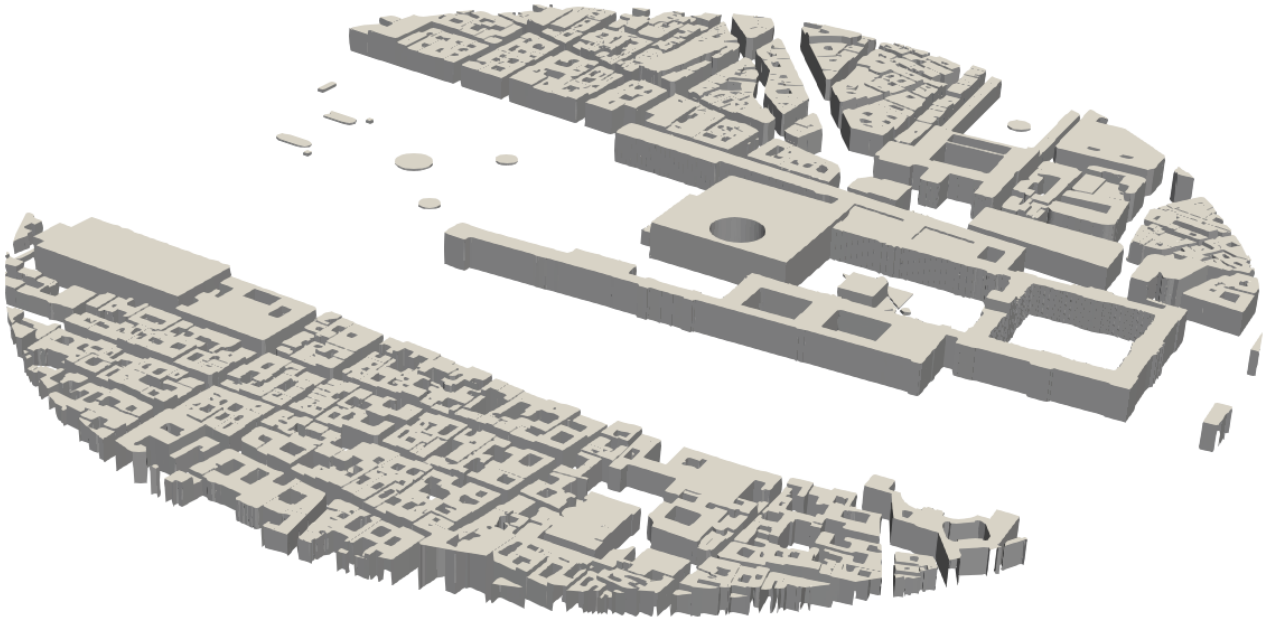


Using polyprep.py

https://github.com/tudelft3d/City4CFD/wiki/Polygons#polygon-generalisation

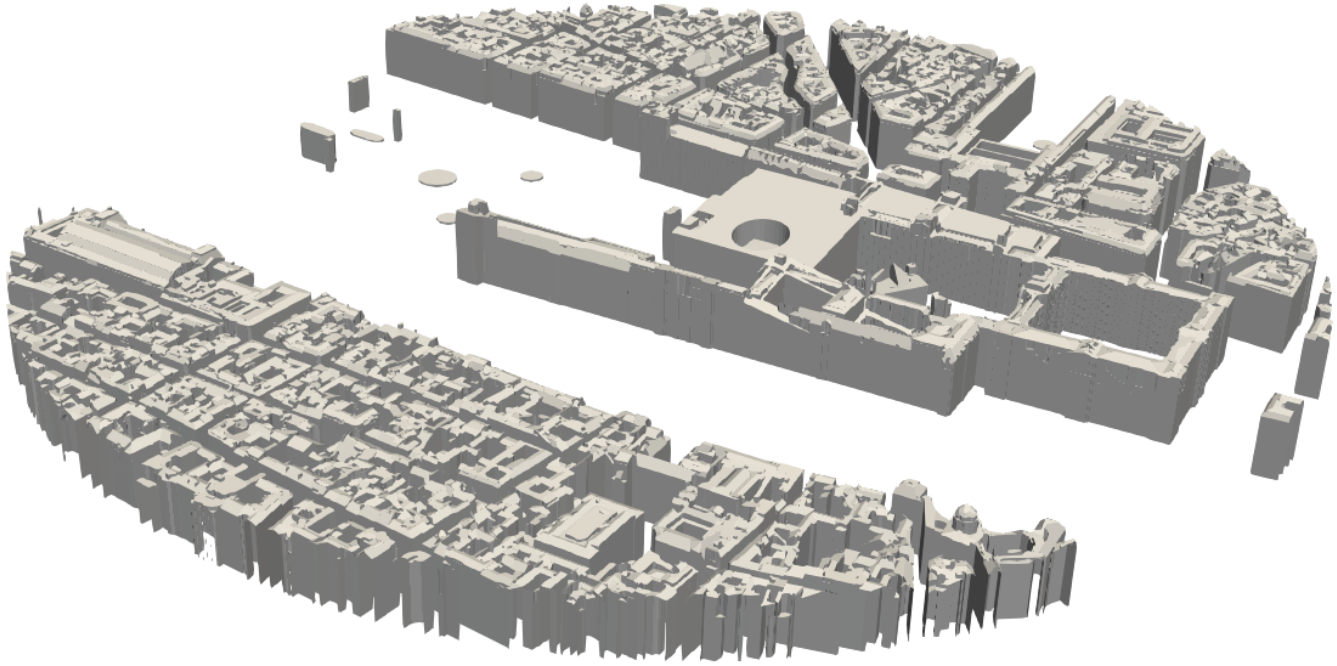


Paris - Reconstruction



LoD 1.2

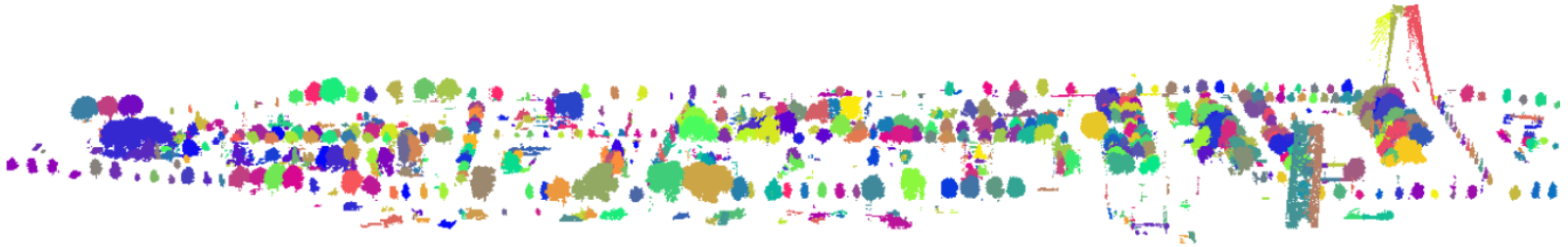
LoD 2.2

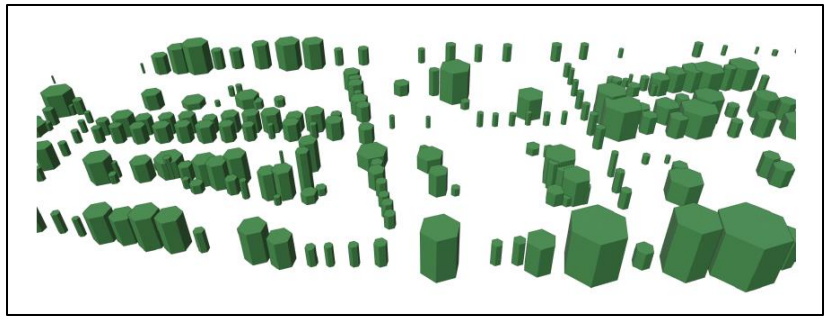


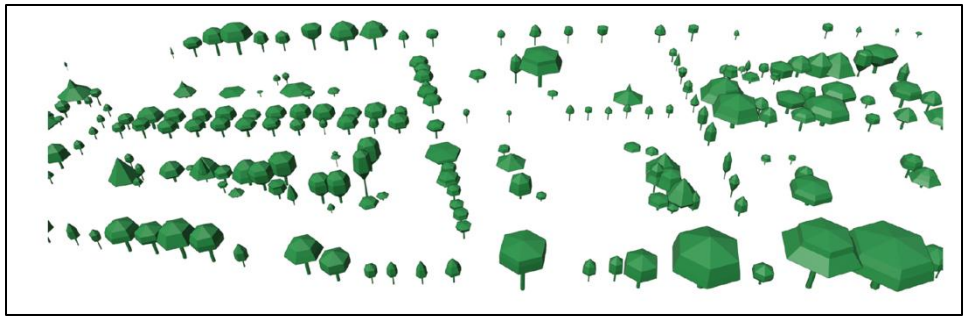
Appendix

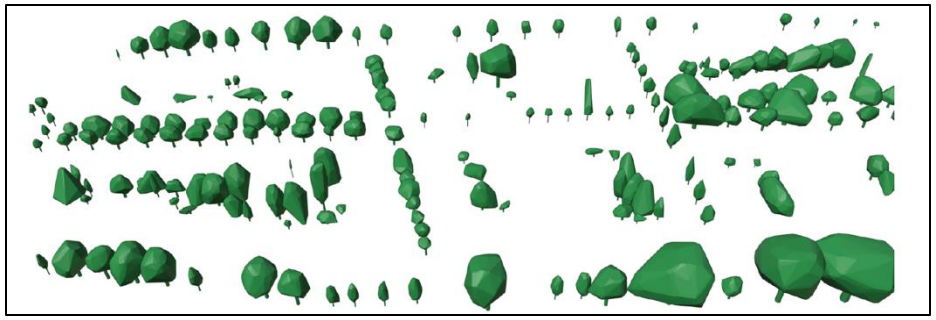
What we want to work on

Tree reconstruction from point clouds

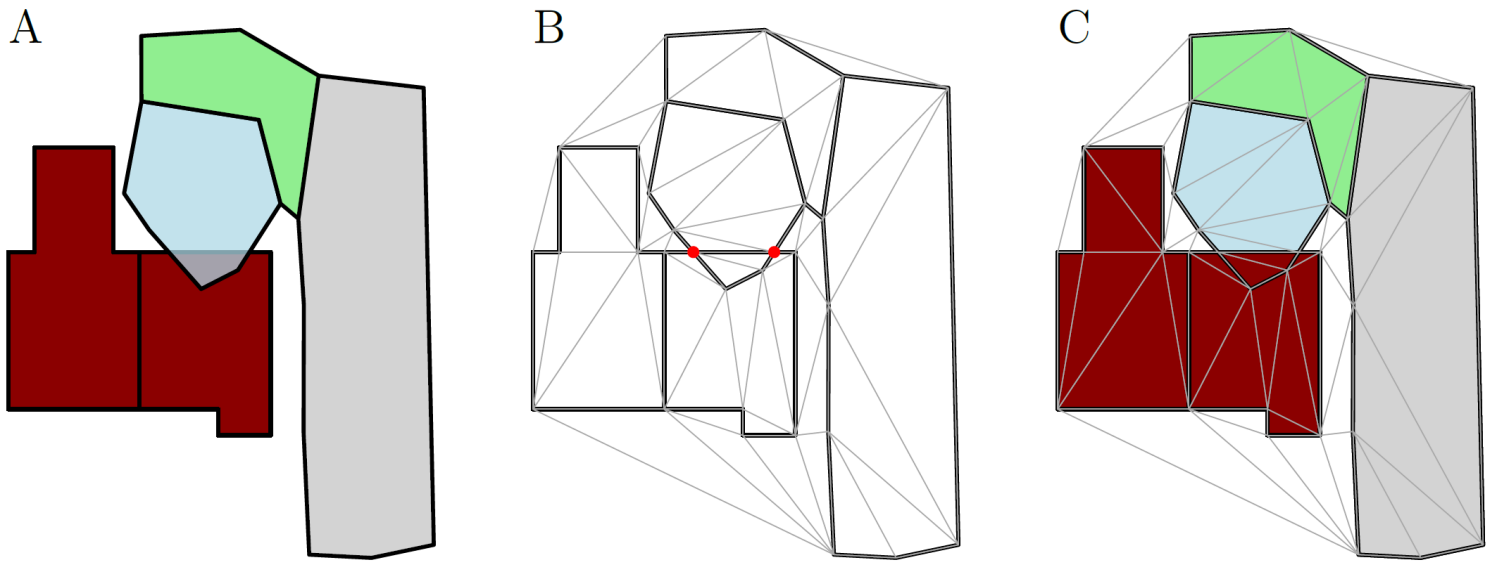








Constrained triangulation and conflict solving



List of commands used


```
7594  python3 -m venv venv
7596  source venv/bin/activate
7598  python3 polygons/fetch.py
7599  pip install osmnx
7600  cd polygons
7603  python3 polygons/fetch.py
7604  python3 fetch.py
7605  pip install geopy
7606  python3 fetch.py
7608  python3 fetch.py > log.fetch
7609  vi log.fetch
7610  ls data
7611  cd ..
7614  cd PC
7615  ls
7617  lasmerge64 -i *.LAZ -keep_class 2 -o ground.laz
7618* lasmerge64 -i *.LAZ -keep_class 6 -o buildings.laz
7620  cd ../
7621  cd reconstruction
7623  vi ../polygons/log.fetch
7625  mv results results_LoD2.2
7626  city4cfd rotterdam_LoD2.json --output_dir results_LoD2.2
7630  deactivate
```