GEOProcessing 2025

Presenter: Dr. Mohamed El-Darieby

A Workflow for Map Creation in Autonomous Vehicle Simulations

Zubair Islam, Ahmaad Ansari, George Daoud, Mohamed El-Darieby

Dr. Mohamed El-Darieby

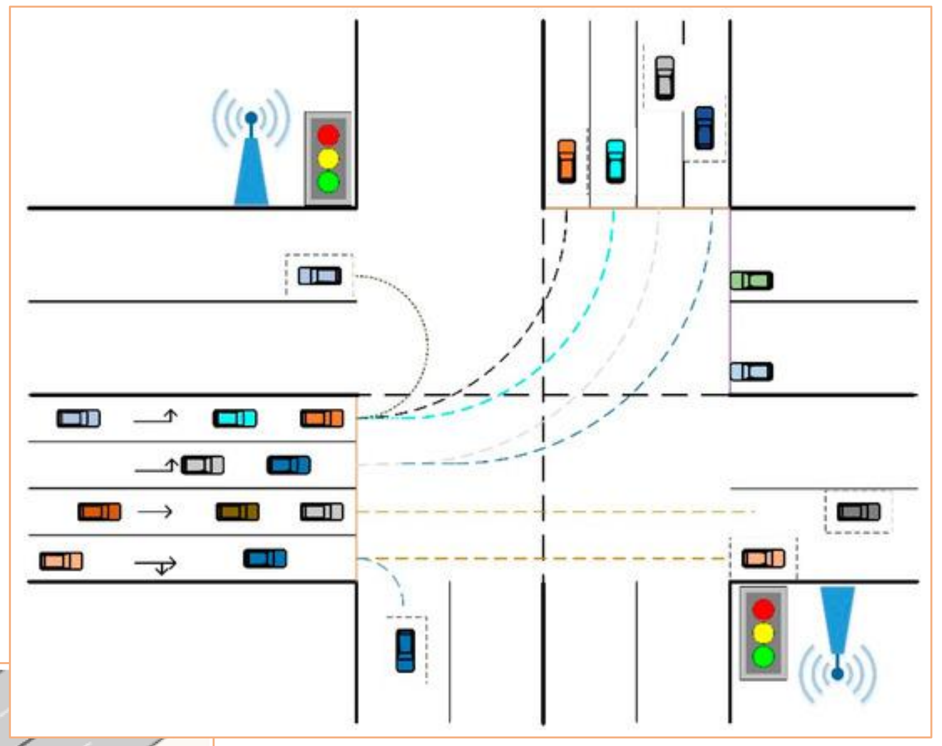
- Associate Professor & Chair
  - Electrical, Computer & Software Engineering at Ontario Tech University, Canada
  - **Email:** mohamed.el-darieby@ontariotechu.ca

- Research Interest:
  - Deep Learning for Autonomous Systems.
  - Vehicular Data
  - Intelligent Transportation Systems

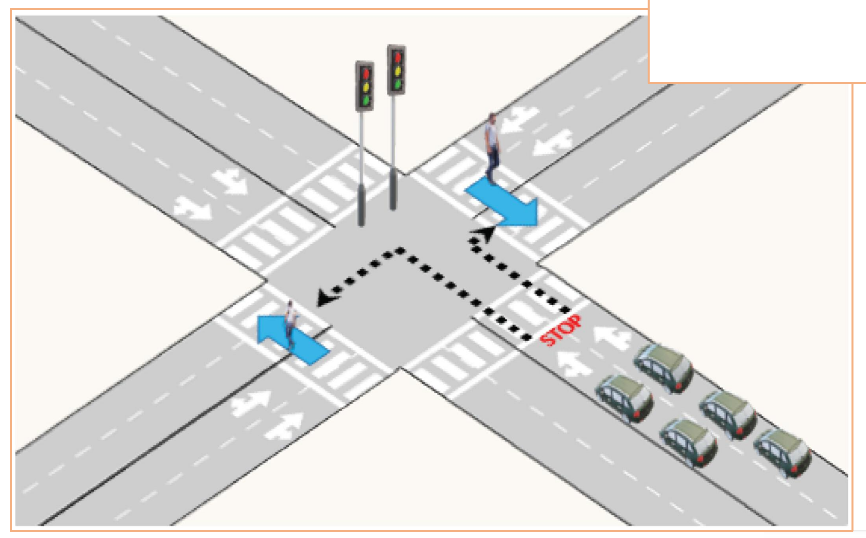
Mohamed El-Darieby
Ontario Tech University, Canada

Impact and Importance

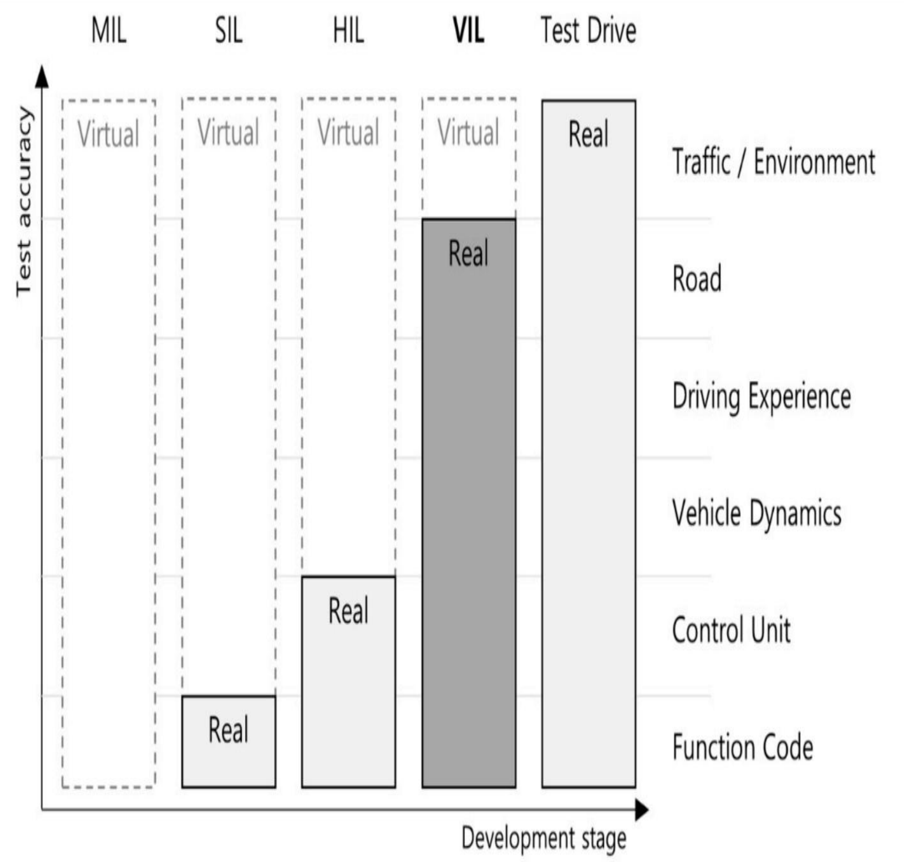
- AV Software Stack
  - Perception
  - Localization
  - Prediction of travel trajectory and planning future movement
- AV Development
  - Need to create tailored simulated environments
  - Safety Assurance
    - Millions (if not more) of testing scenarios
    - Simulation/software in the loop.
      - Hardware/vehicle in the loop



[1] Multi-lane intersection layout with AVs and traffic flow visualization.



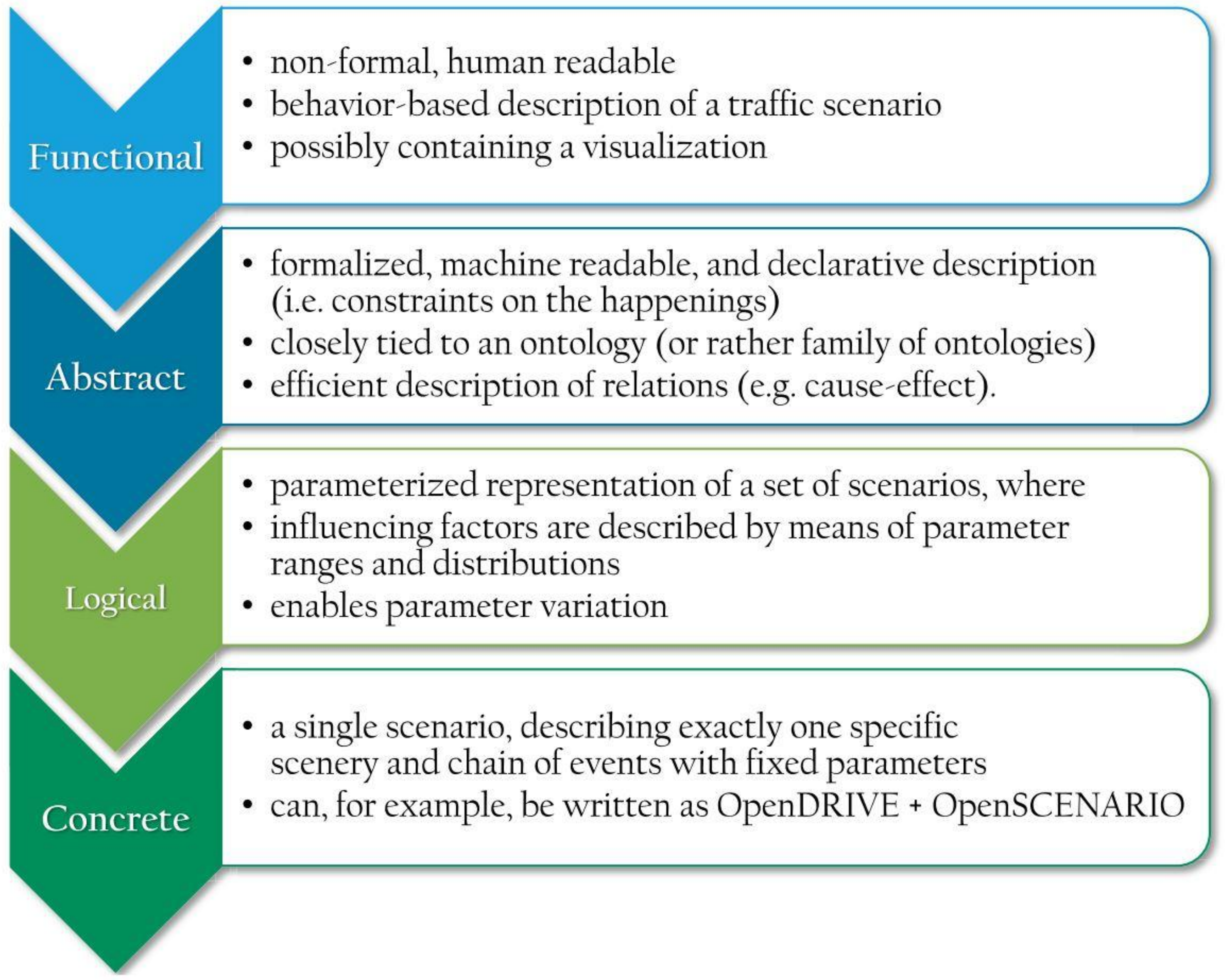
[2] Fuzzy logic-based path planning at an intersection.



[3] Scenario-based testing architecture for automated driving systems.

HD Maps & AV

- Requirements
  - Essential
  - Accuracy
    - Centimeter-level
  - Reality
  - Flexibility in supporting driving scenarios
- Roads
  - Layouts
    - Lane markings
    - Traffic infrastructure
- Road agents
  - Other cars,
  - Pedestrians, ..etc



[4] Key terms in OpenSCENARIO DSL for AV simulation.

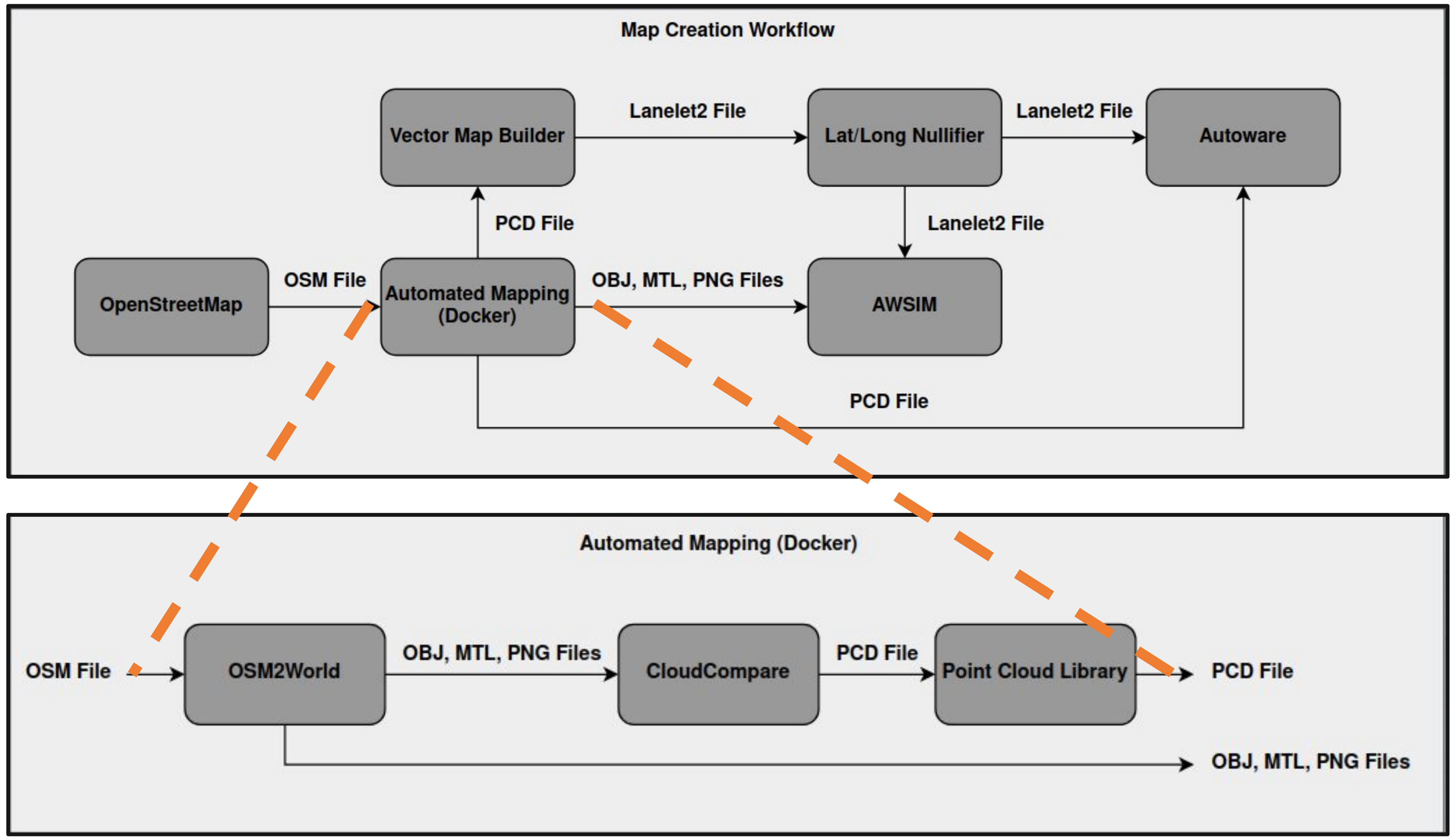
Related Work

- Simulators
  - (e.g., AWSIM, CARLA, LGSVL)
  - Offered only a single simulation map, which represented a large city environment.
  - Documentation/ Instruction given for manually creating custom environments
- Works with Complex Simulation Setup
  - Feng (2020): converts OSM data into simulation-ready maps for CARLA and LGSVL
  - Santonato (2020): Focused on CARLA; limited in compatibility with other simulators
  - Jeong (2022): HD map generation using LiDAR/GNSS data; scalability issues
- Works Requiring Expensive Real-World Data and AI Development
  - Li (2022): A deep learning-based method for HD map generation, offering high accuracy

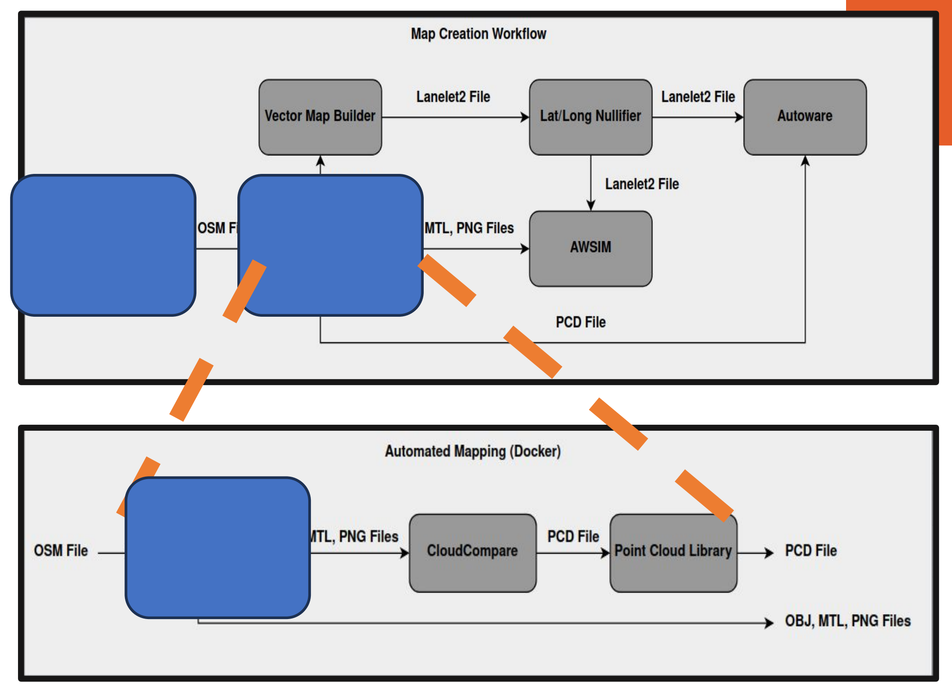
Contribution

- A Workflow That:
  - **Simplifies & streamlines** the process of generating simulation-ready maps.
  - Creates environments as close to real-world as possible
    - Adaptable to any location available on OpenStreetMap (OSM)
    - Enables more realistic testing environments.
  - Uses open-source tools and minimal computational resources
- Enables Integration with AV Simulation Platforms:
  - **With AWSIM**: Generate a Lanelet2 OSM map and 3D mesh
  - **With Autoware Integration**: Produce Lanelet2 map and point cloud

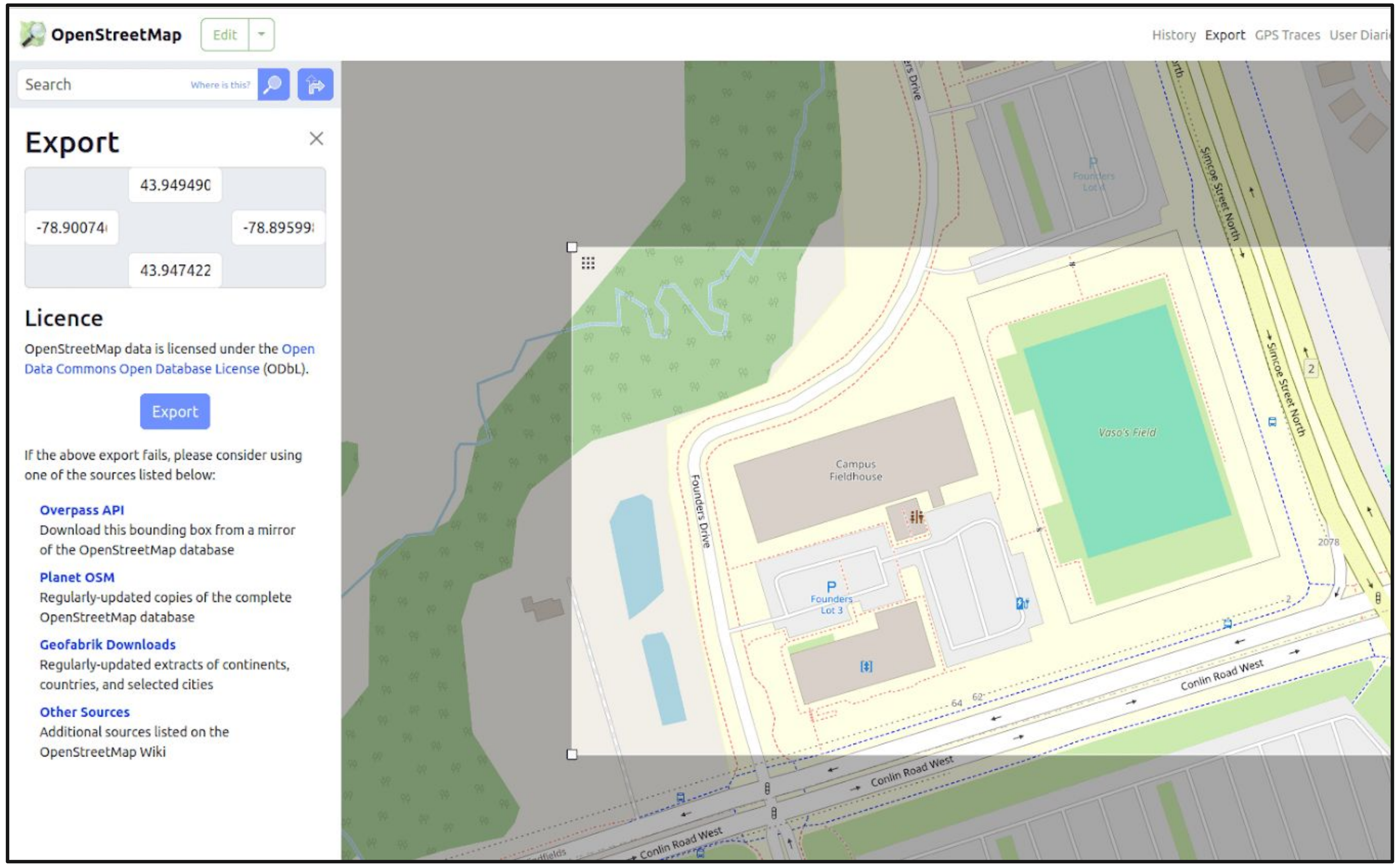
3D Map Creation Workflow



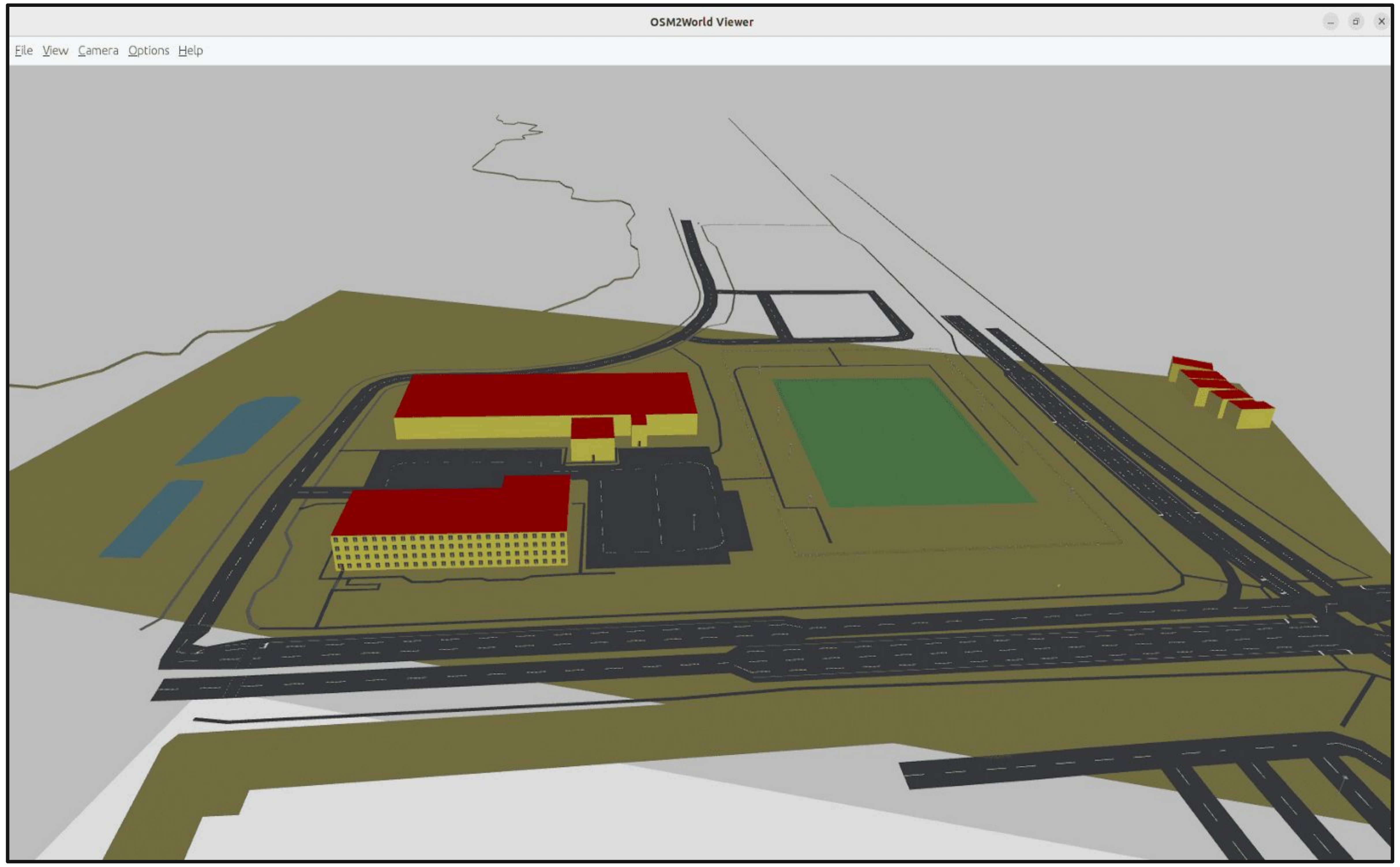
Workflow is available on Github [5]



OSM File to 3D Mesh

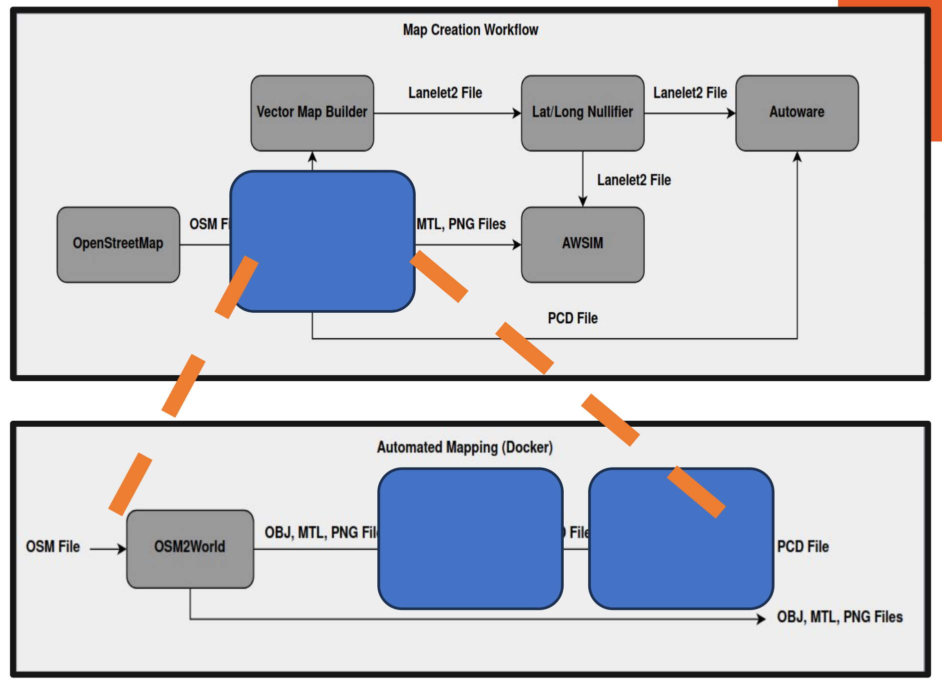


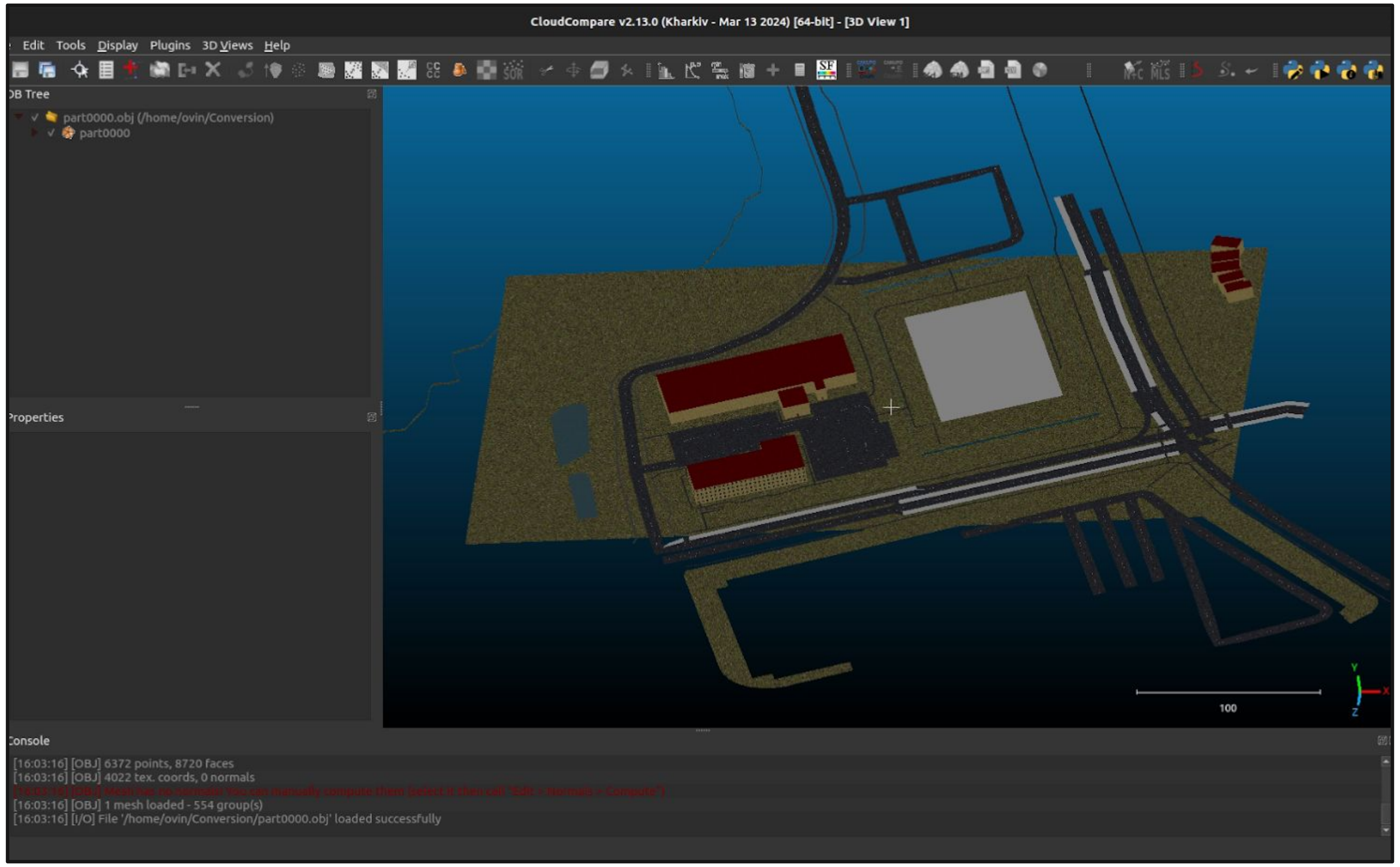
Picking a location in **OpenStreetMap [6]**
Exported as: *OSM File*



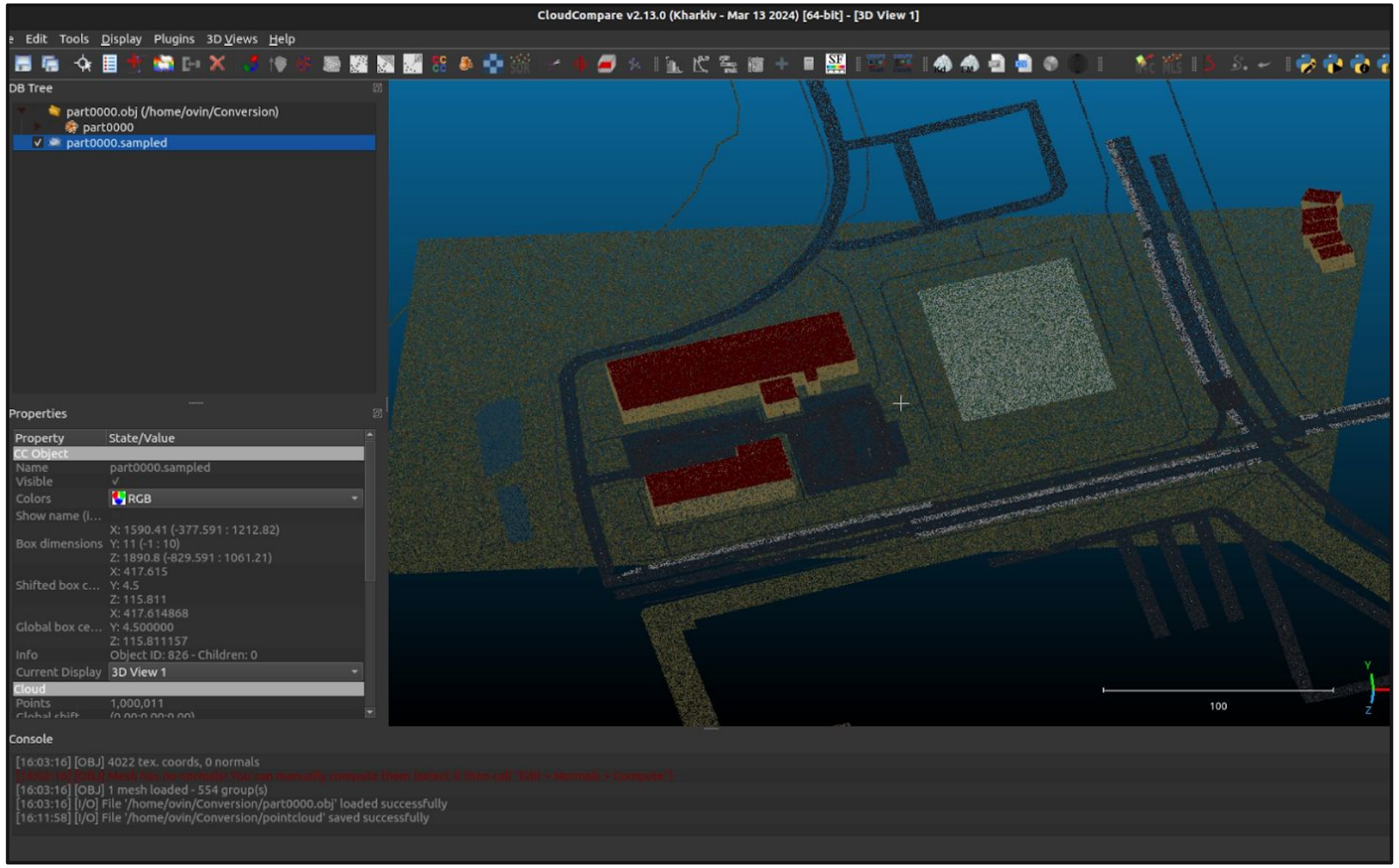
Docker Container: OSM file converted into a 3D mesh in **OSM2World [7]**
Exported as: *3D map files (OBJ, MTL, PNG)*

Point Cloud Generation



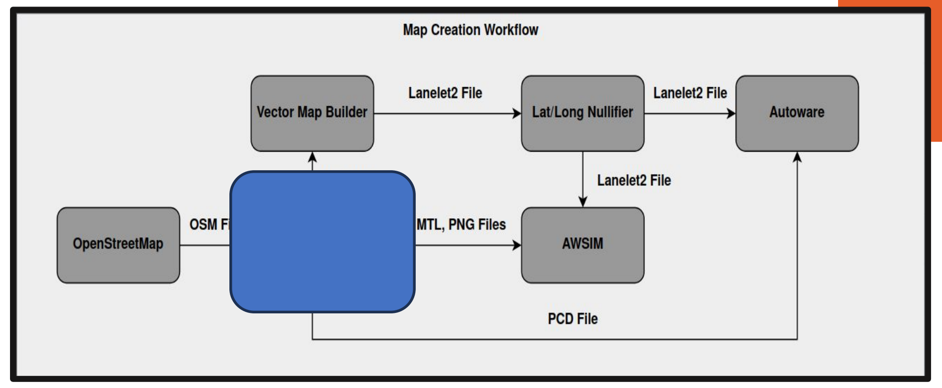


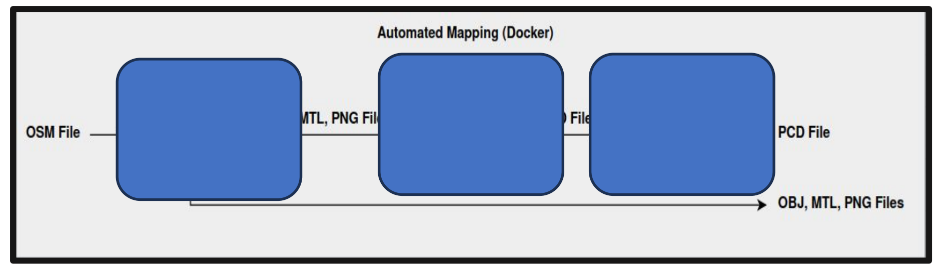
Docker Container: 3D mesh loaded into **CloudCompare [8]**



Docker Container: 3D Mesh Converted into Point Cloud

Exported as: *Point Cloud Data File*

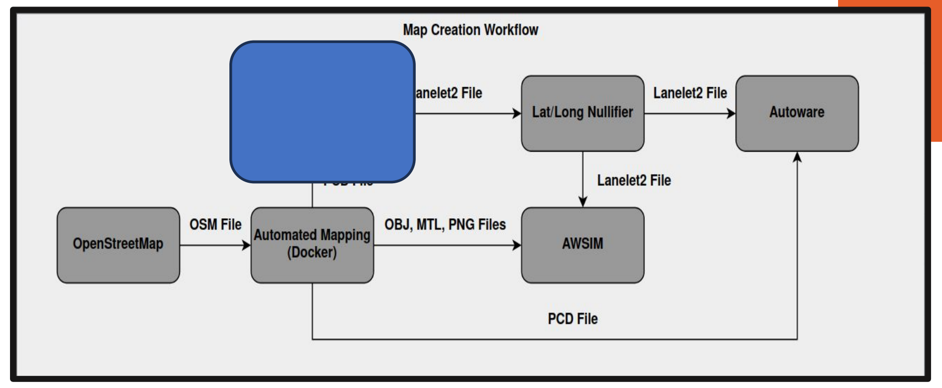


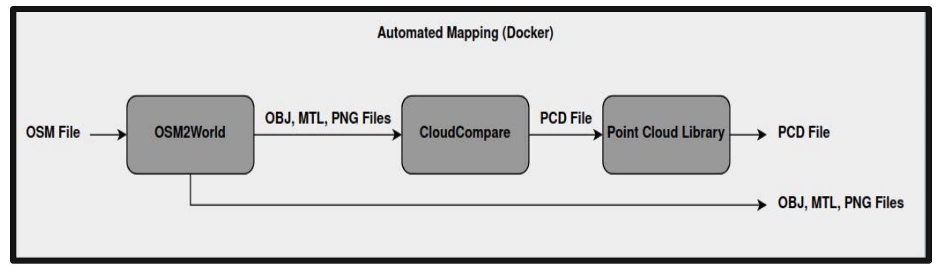


Point Cloud Processing

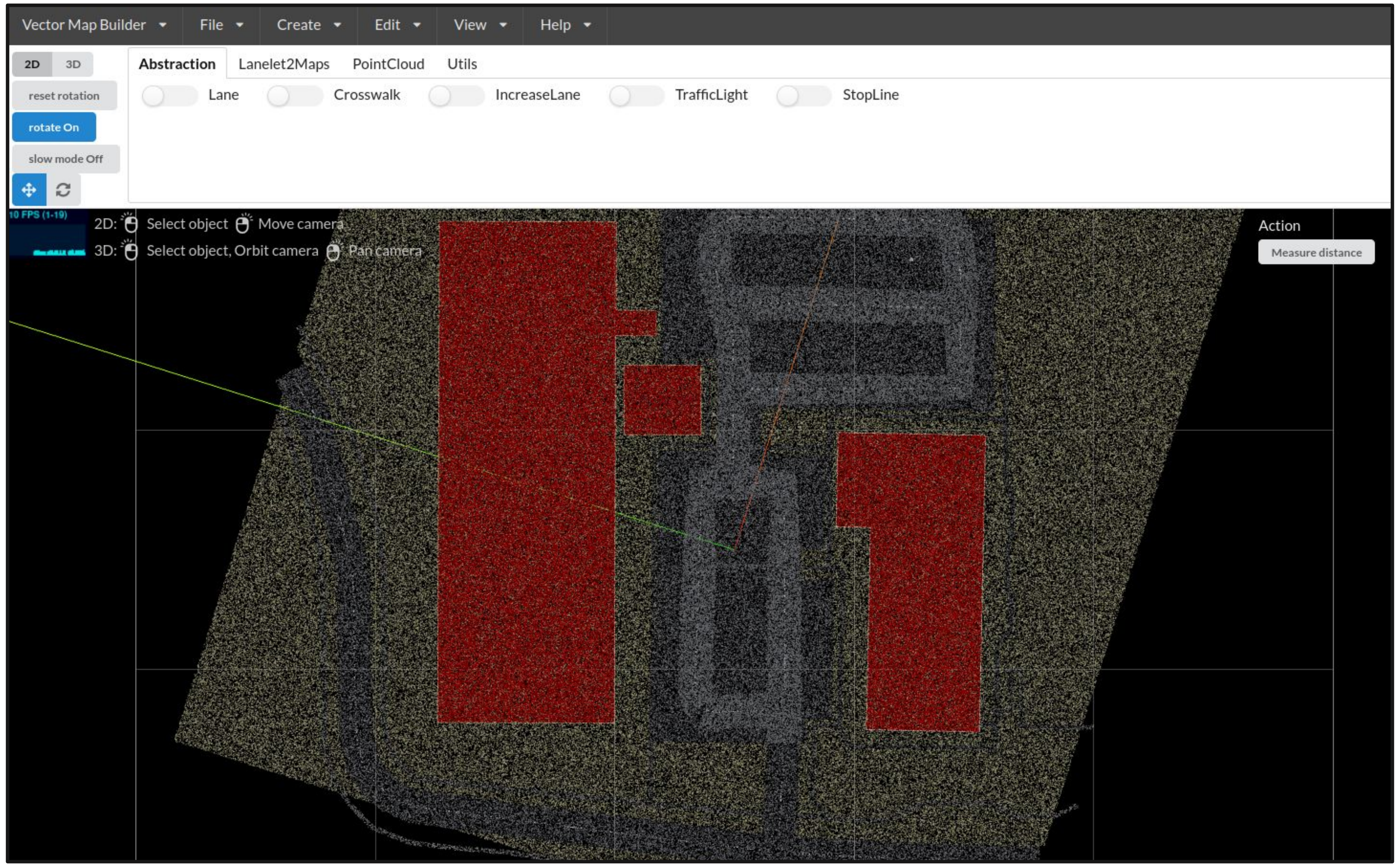
Docker Container: Point Cloud loaded into **Point Cloud Library (PCL) [9]** with a frontal view

Docker Container: Point Cloud after further processing using **PCL**, resulting in a top-down aerial view
Exported as: *Point Cloud Data File*

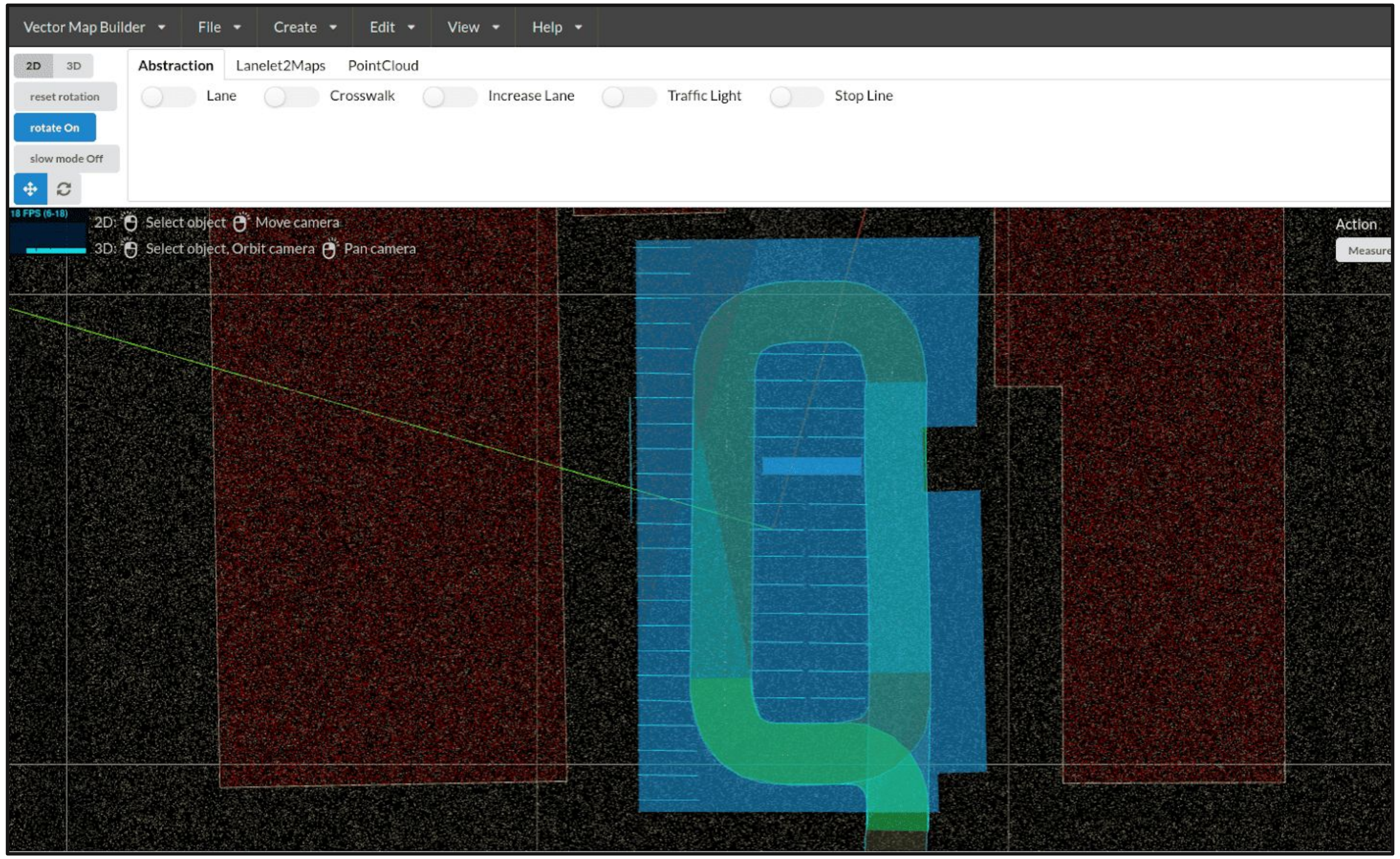




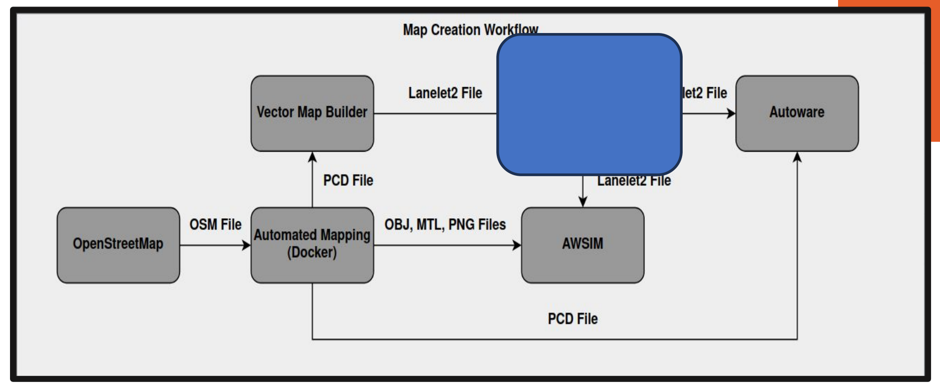
Lanelet2 OSM File Creation

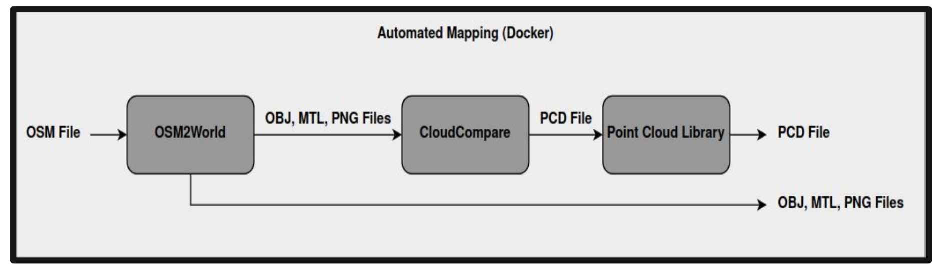


Point Cloud loaded into **Vector Map Builder [10]**



Manually creating lanelets and parking spaces
Exported as: *Lanelet2 OSM File*





Latitude/Longitude Nullifier Script

- This script [11] removes latitude and longitude fields from the Lanelet2 OSM file to prevent map issues in Autoware and AWSIM, ensuring the lanes don’t extend beyond the map boundaries.

```python
import sys
import xml.etree.ElementTree as ET

def update_lat_lon_to_empty(node):
    # Set lat and lon attributes to empty strings
    node.set("lat", "")
    node.set("lon", "")

def update_lat_lon_in_file(input_file, output_file):
    # Parse the XML file
    tree = ET.parse(input_file)
    root = tree.getroot()

    # Iterate over all elements and update lat and lon if both attributes are present
    for elem in root.iter():
        if 'lat' in elem.attrib and 'lon' in elem.attrib:
            update_lat_lon_to_empty(elem)

    # Save the modified XML to the output file
    tree.write(output_file, encoding="utf-8", xml_declaration=True)

def main():
    # Check if the correct number of command line arguments is provided
    if len(sys.argv) != 3:
        print("Usage: python script.py input_file.xml output_file.xml")
        sys.exit(1)

    # Get input and output file names from command line arguments
    input_file = sys.argv[1]
    output_file = sys.argv[2]

    # Update lat and lon in the input file and save to the output file
    update_lat_lon_in_file(input_file, output_file)

if __name__ == "__main__":
    main()
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<osm generator="VMB">
  <MetaInfo format_version="1" map_version="2" validation_version=""/>
  <node id="1" lat="43.34567092534" lon="-79.76723736903">
    <tag k="local_x" v="-84.195"/>
    <tag k="local_y" v="-60.0152"/>
    <tag k="ele" v="-0.0002"/>
  </node>
  <node id="2" lat="43.34575688904" lon="-79.76688411699">
    <tag k="local_x" v="-55.7046"/>
    <tag k="local_y" v="-50.0454"/>
    <tag k="ele" v="-0.0002"/>
  </node>
```

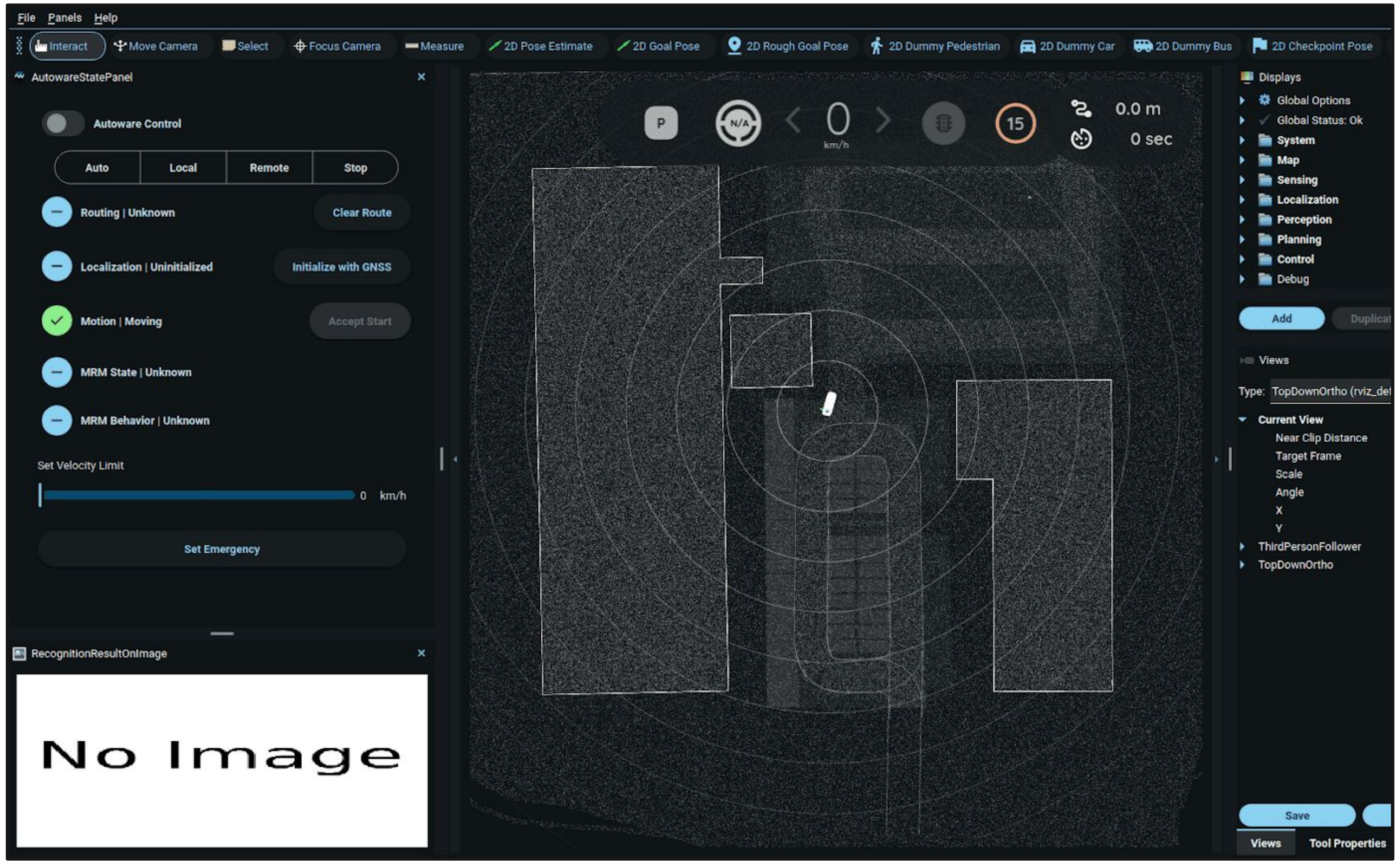
Before

```xml
<?xml version='1.0' encoding='utf-8'?>
<osm generator="VMB">
  <MetaInfo format_version="1" map_version="2" validation_version="" />
  <node id="1" lat="" lon="">
    <tag k="local_x" v="-84.195" />
    <tag k="local_y" v="-60.0152" />
    <tag k="ele" v="-0.0002" />
  </node>
  <node id="2" lat="" lon="">
    <tag k="local_x" v="-55.7046" />
    <tag k="local_y" v="-50.0454" />
    <tag k="ele" v="-0.0002" />
  </node>
```

After

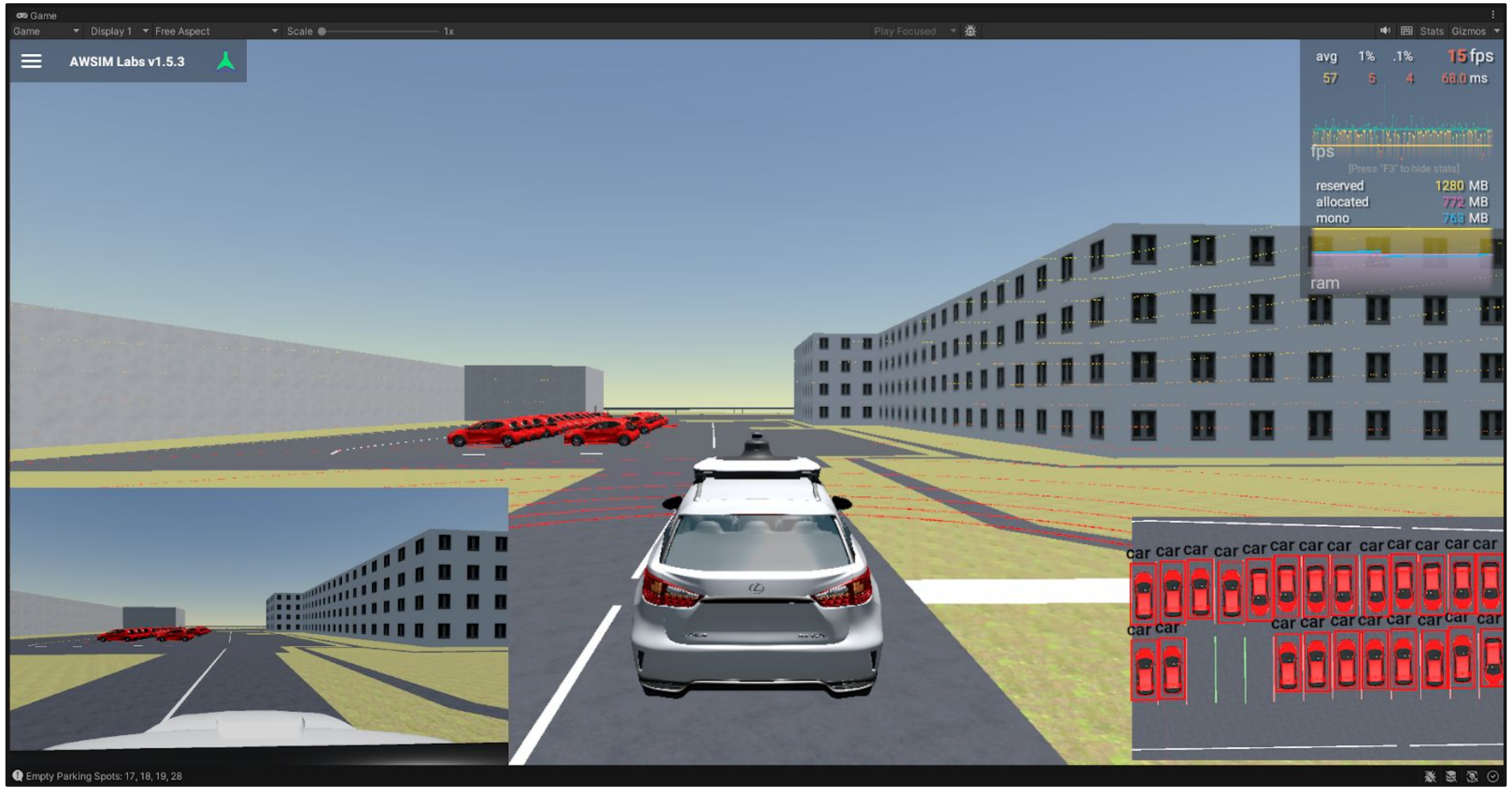
Loading the Environment Into Autoware and AWSIM

Lanelet2 map and 3D model loaded in **AWSIM [12]**

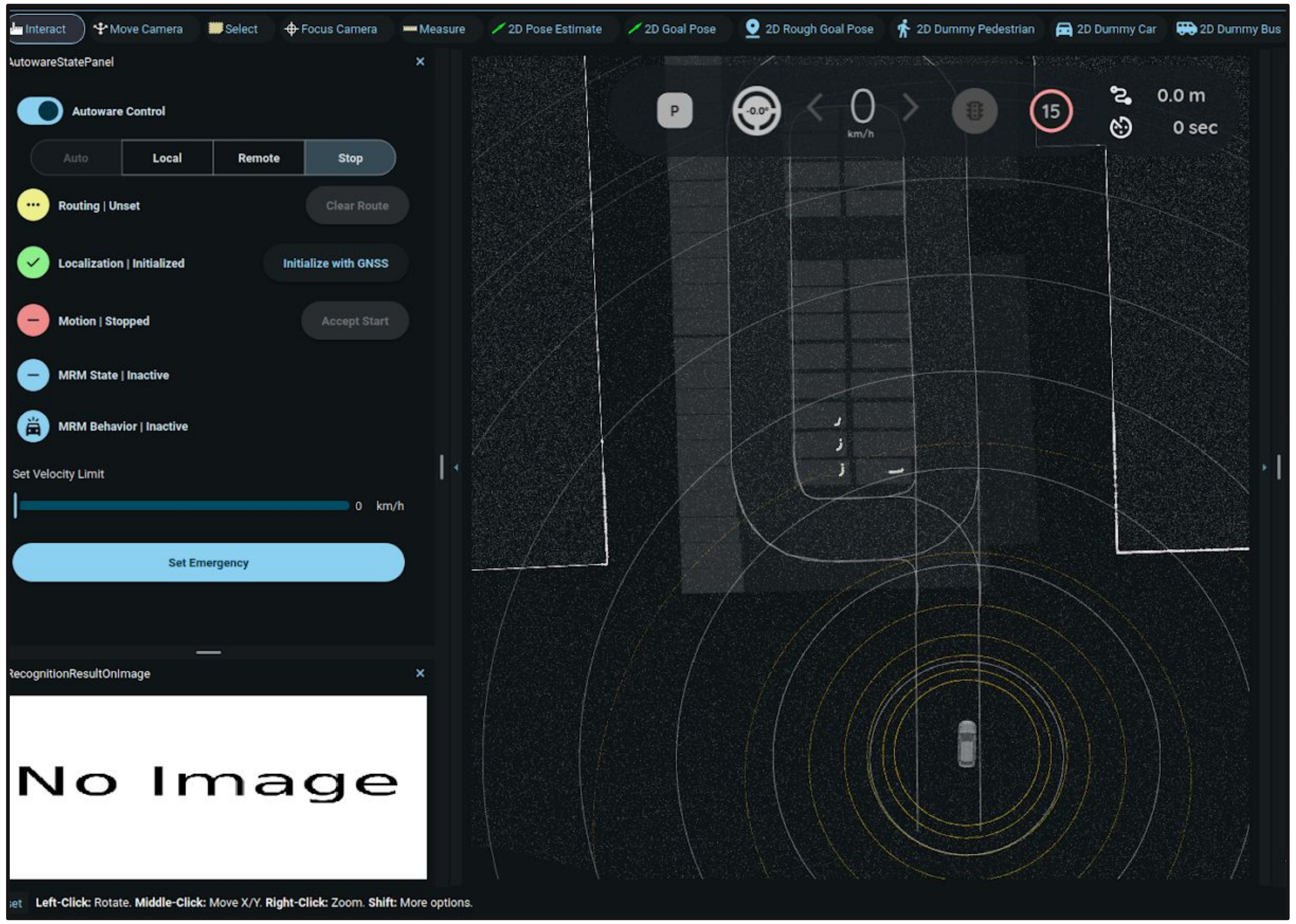


Point Cloud and Lanelet2 loaded in **Autoware [13]**

Results: Initial Startup of AWSIM and Autoware

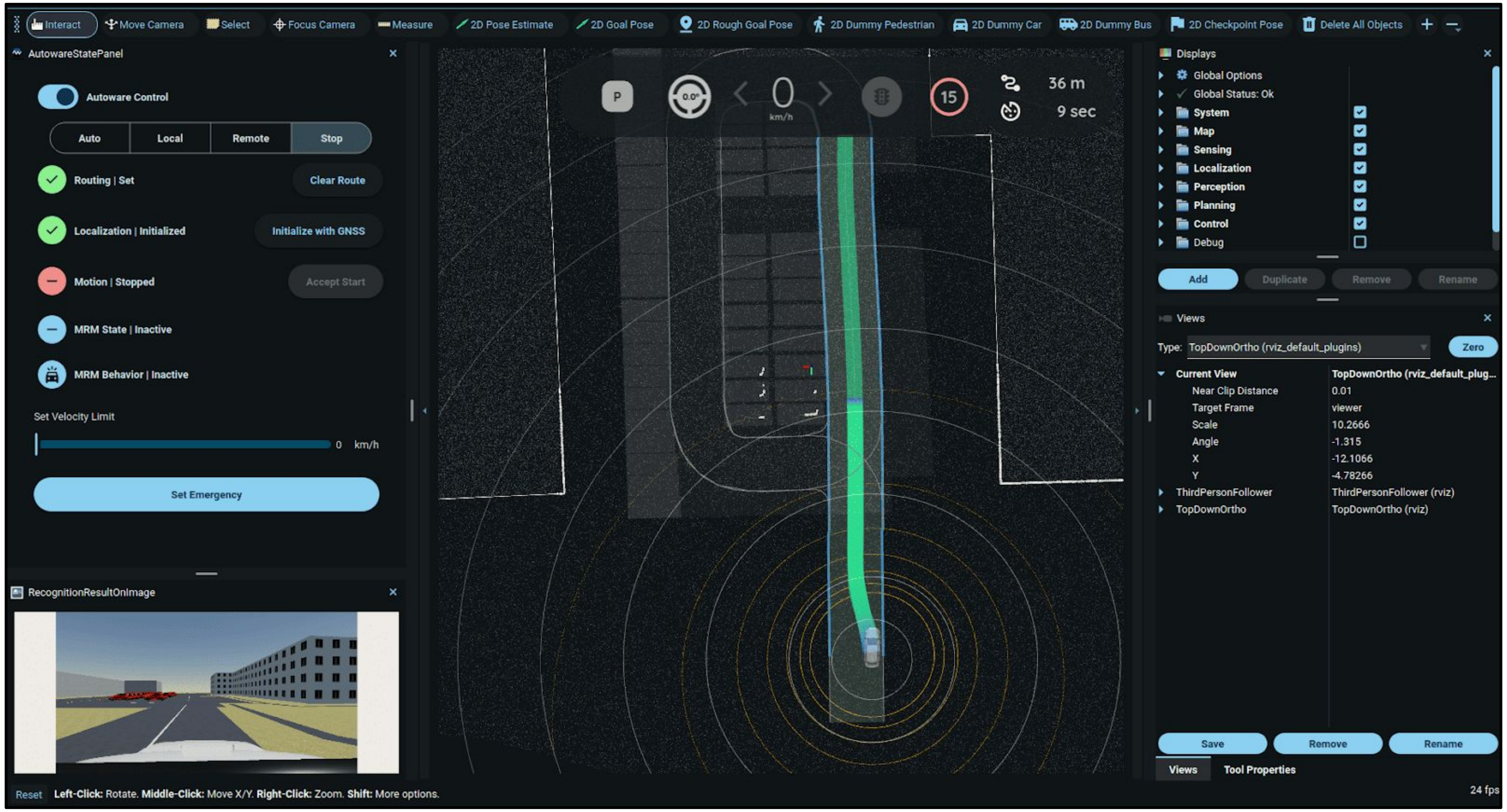


AWSIM is started first, sending all sensor data to **Autoware**



Vehicle localizes in **Autoware**, after receiving data from **AWSIM**

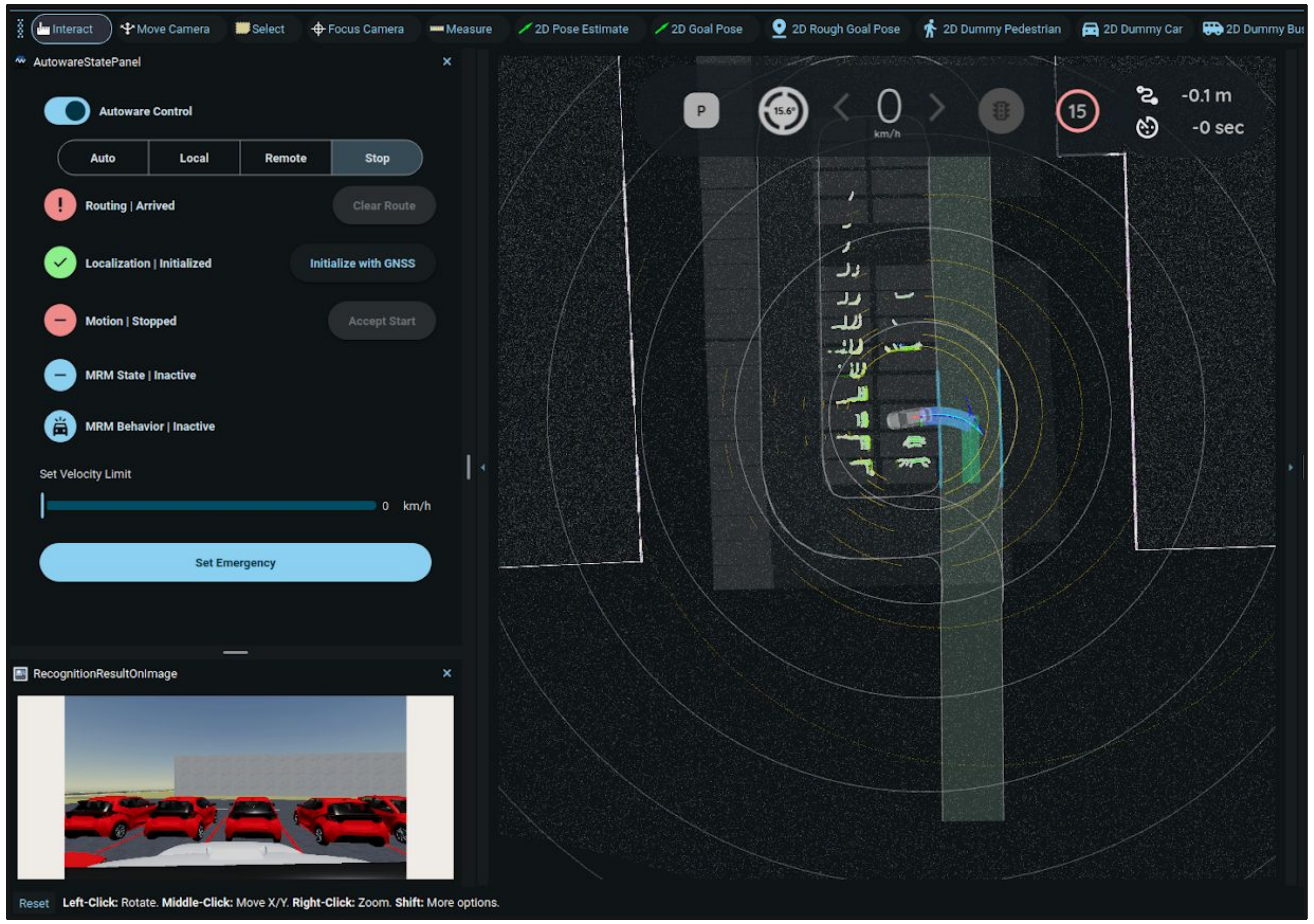
Results: Planning a Parking Scenario using Autoware



After setting a goal pose in a parking spot, **Autoware** calculates the route

Results: Parking Scenario Completed

Car reaches goal in **AWSIM**



Car reaches goal in **Autoware**

Results Conclusion

- Successful Simulation Testing:
  - The workflow enables accurate AV testing in simulation environments, successfully completing a parking scenario in both AWSIM and Autoware.
- Real-World Potential:
  - The streamlined process demonstrates the ability to generate simulation-ready maps for various AV testing environments, enabling realistic testing and development.
  - Workflow is adaptable to various scenarios (e.g., parking lots, urban streets).
  - Reduces barriers to entry for researchers and developers, making AV simulation more accessible.

Future Work

- Improved Model Accuracy
  - Enhancing the precision of generated maps to better represent real-world environments.
- Integration of SLAM Technologies
  - Using SLAM to generate accurate point clouds for real-world deployment, addressing limitations in current 3D model accuracy.
- Optimizing Workflow Compatibility
  - Expanding the workflow’s compatibility with a broader range of simulators, ensuring flexibility for diverse testing environments.
- Flexible Handling of Latitude and Longitude Values
  - Improve control over the nullification process to enhance map accuracy

References

[1] “A Typical Multi-Lane Intersection Scenario,” ResearchGate, 2021. https://www.researchgate.net/figure/A-typical-multi-lane-intersection-scenario_fig1_352269970
[2] Zhang et al., “Path Planning for AVs Using Fuzzy Logic,” Wiley, 2023. https://doi.org/10.1002/asmb.2772
[3] Gamba et al., “Automated Driving Test Scenarios,” MDPI, 2023. https://www.mdpi.com/2624-8921/6/2/44
[4] ASAM OpenSCENARIO, “Key Terms & Conceptual Overview,” 2025. https://publications.pages.asam.net/standards/ASAM_OpenSCENARIO/ASAM_OpenSCENARIO_DSL/latest/conceptual-overview/key_terms.html
[5] “OSM to Pointcloud and Lanelet Conversion Process,” GitHub, https://github.com/zubxxr/OSM-to-Pointcloud-and-Lanelet-Conversion-Process (accessed 2025-02-15).
[6] OpenStreetMap, https://www.openstreetmap.org/ (accessed 2025-02-15).
[7] T. Knerr, “OSM2World create 3D models from OpenStreetMap,” OSM2World, https://osm2world.org/ (accessed 2025-02-15).
[8] “Presentation,” CloudCompare, https://www.cloudcompare.org/presentation.html (accessed 2025-02-15).
[9] “About,” Point Cloud Library, https://pointclouds.org/about/ (accessed 2025-02-15).
[10] Vector Map Builder, https://tools.tier4.jp/vector_map_builder_ll2/ (accessed 2025-02-15).
[11] “remove_lat_lon.py,” GitHub, https://github.com/zubxxr/OSM-to-Pointcloud-and-Lanelet-Conversion-Process/blob/main/remove_lat_lon.py (accessed 2025-02-15).
[12] tier4, “Tier4/AWSIM: Open source simulator for self-driving vehicles,” GitHub, https://github.com/tier4/AWSIM (accessed 2025-02-15).
[13] “Home Page,” Autoware, https://autoware.org/ (accessed 2025-02-15).