Ostfalia

Hochschule für angewandte Wissenschaften

Hochschule Braunschweig/Wolfenbüttel

Fakultät Maschinenbau

IMEC - Institut für Mechatronik

Bachelorarbeit

Development of a toolchain for the automated generation of digital maps for autonomous drive

Sergio Peral Garijo

Matrikelnummer: 70483485

Erste Prüferin: Prof. Dr.-Ing. Xiaobo Liu-Henke

Zweiter Prüfer: Prof. Dr.-Ing. Christoph Hartwig

Betreuer: Marian Göllner, M. Eng. & Taihao Li, M. Eng.

Abgabe: 05.03.2024

Declaration

I hereby declare that I am the sole author and composer of my thesis and that no other sources or learning aids, other than those listed, have been used. Furthermore, I declare that I have acknowledged the work of others by providing detailed references of said work.

I hereby also declare that my Thesis has not been prepared for another examination or assignment, either wholly or excerpts thereof.

X _____

Signature

Abstract

This thesis presents a comprehensive exploration into the development of a toolchain designed for the automated generation of digital environments tailored for autonomous driving scenarios. Focused on the road network surrounding Ostfalia Hochschule University as a case study, the aim of this research is to create a digital representation that serves as a reliable foundation for testing and refining autonomous vehicle algorithms.

The toolchain integrates various data sources, including OpenStreetMap for the geographical data and GeoTIFF for the elevation data. Through specific designed modules, the converter translates both geospatial data into the OpenDrive format.

The thesis contributes to the field by expanding the capabilities of autonomous vehicle testing through automated digital environment generation. It bridges the gap between geospatial data sources and advanced simulation requirements, facilitating a seamless transition from raw data to a dynamic digital environment.

By providing a foundation for comprehensive and realistic simulations, this tool chain contributes to the advancement of safe and efficient autonomous driving solutions.

Keywords:

Autonomous driving, Toolchain development, Digital mapping, Automation, Sensor fusion, Machine learning, Image processing, Computer Vision, LiDAR, Radar, Localization, Mapping algorithms, Road network extraction, Data integration, High-definition maps

Contents

List of abbreviations

Abbreviation	Full description
API	Application Programming Interface
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
DOF	Degrees of Freedom
DEM	Digital Elevation Model
GPS	Global Positioning System
GDPR	European Union's General Data Protection Regulation
GNSS	Global Navigation Satellite System
HD map	High-Definition map
IMU	Inertial Measurement Units
LIDAR	Light Detection and Ranging
NASA	National Aeronautics and Space
NASADEM	NASA Digital Elevation Model
MMS	Mobile Mapping Systems
MIMO	Multiple Input Multiple Output
OSM	Open Street Maps
RADAR	Radio Detection and Ranging
SLAM	Simultaneous Localization and Mapping

SRTM	Shuttle Radar Topography Mission
WGS64	World Geodetic System 1984
XML	Extensible Markup Language
SUMO	Simulation of Urban MObility

List of Figures

List of Tables

1. Introduction

Autonomous driving technology requires extensive testing and validation before it can be deployed on public roads. This is necessary to ensure that the system can detect and respond to various driving scenarios and conditions, such as changes in weather, road conditions, and traffic patterns.

However, testing autonomous driving systems in the real world can be time-consuming, expensive, and potentially dangerous. To overcome these challenges, researchers and engineers use digital environments for simulation to test and validate autonomous driving technology. This digital environment provides a virtual platform where researchers can create various driving scenarios and conditions that the autonomous system may encounter in the real world.

In a digital environment, researchers can also control various factors, such as the weather, traffic density, and road conditions, to simulate different scenarios and test the system's response. This allows for more efficient and cost-effective testing, as well as the ability to test the system in a range of scenarios that might be difficult or even impossible to replicate in the real world.

Moreover, the use of a digital environment also enables researchers to collect and analyze large amounts of data generated during the testing process. This data can be used to identify areas where the autonomous system needs improvement and refine the system's performance in real-world situations.

In summary, a digital environment for simulation is essential for testing and validating autonomous driving technology, as it allows for safe, efficient, and cost-

effective testing of autonomous systems in a wide range of driving scenarios and conditions.

1.1 Problem

The problem that this thesis aims to solve is the inefficiency and limitations of current methodologies for creating realistic models for road traffic from open-source data.

Manual creation of these models is time-consuming, error-prone, and requires significant technical expertise. Additionally, traditional 2D modeling techniques lack the detail and accuracy of reality required for the development and testing of autonomous vehicles.

Therefore, there is a need for an automated tool chain that can create highly accurate and detailed 3D environment models for road traffic in a more efficient and cost-effective manner. This will enable the development and testing of autonomous vehicles to be conducted with greater accuracy and reliability, leading to improved safety on our roads.

Level of detail

The level of detail of the OpenStreetMap related to specific features such as the lanes, junctions or elevation is sometimes so poor that can be a problem for autonomous driving.

This insufficient information about complex roads may be difficult for autonomous vehicles to accurately interpret and navigate in those situations, potentially leading to incorrect maneuvers or unsafe behavior.

Elevation in map dataset

The open-source map dataset provided by the OpenStreetMap community typically lacks elevation information, although it can be manually added using the "ele=*" tag. [1] Despite this augmentation, the depiction of elevation in these maps frequently falls short of realism due to their node-based mapping structure. This approach depicts roads as connections between discrete nodes, resulting in a spiky and uneven representation of elevation.

This presents a notable limitation, as accurate elevation data is crucial for various applications. To overcome this challenge, reliance on external datasets containing elevation information becomes necessary.

Transitioning to the OpenDrive format doesn't entirely alleviate the need for external elevation datasets. However, it offers a more promising solution by utilizing geometrical representations of roads. [2] This results in a more realistic portrayal of elevation, facilitating smoother transitions and a more accurate depiction of terrain variations.

Elevation data errors

One of the primary challenges encountered when utilizing open-source elevation data lies in its lack of accuracy, particularly in generating suitable digital environments for autonomous vehicle simulation. The highest resolution offered by any open-source solution typically stands at 1 arc second, translating to a horizontal resolution of 30 meters. [3] However, this resolution often proves insufficient, particularly in areas with significant elevation variations.

This limitation can result in undesirable errors, especially on uneven terrains where large elevation changes occur. As a consequence, relying solely on open-source elevation data for autonomous vehicle simulation may lead to inaccuracies that compromise the effectiveness and reliability of the simulation environment.

1.2 Contribution

The proposed toolchain will reduce the time and resources required for the creation of environment models and will enable non-technical personnel to participate in the modeling process due to the simplicity of the tool.

To accomplish this goal, it will use the existing open-source tools as the osm2xodr converter [4] from GitHub developed by Jan-Hendrik Meusener and make some modifications to get a proper digital environment for autonomous driving. The contribution to this tool is to modify the generation of the elevation data because the method used for the generation of the elevation was an unrealistic approach bearing in mind that the source file for the elevation data was a png, with all the limitations that this kind of format has for the storage of such a complex representation.

Furthermore, the toolchain makes a significant contribution by developing a methodology for implementing elevation data in digital environments. This methodology involves generating a continuous surface that incorporates accurate and detailed elevation information.

2. Objectives

The objectives of this bachelor thesis project are as follows:

1. Conduct a thorough literature review to identify existing methodologies for creating environment models for road traffic and determine areas where automation can be introduced.
2. Develop a tool chain that automates the creation of digital map models for autonomous drive.
3. Conduct testing and validation of the tool chain to ensure its accuracy, reliability, and scalability.
4. Evaluate the effectiveness of the tool chain in creating accurate and high-quality environment models for road traffic.
5. Analyze the advantages and limitations of the proposed tool chain compared to existing methodologies.
6. Document the development process, including technical specifications, implementation details, and testing procedures.
7. Demonstrate a comprehensive understanding of the field and make recommendations for future research in automated environment modeling for road traffic.

3. Background and motivation

The development of autonomous driving technologies has gained significant attention and momentum in recent years. [5] The automotive industry has been investing heavily in research and development efforts to advance the capabilities and safety of autonomous vehicles. One critical aspect of autonomous driving is the accurate representation of the surrounding environment, particularly road traffic conditions. Creating environment models that capture the complex dynamics and interactions of various road users is essential for the successful deployment of autonomous vehicles.

The above challenges and the growing demand for autonomous driving solutions have created a pressing need for an efficient and automated tool chain for the creation of environment models for road traffic. By developing such a tool chain, several benefits can be realized:

- Efficiency and Cost Reduction: Automating the process of environment model creation will significantly reduce the time and effort required, allowing for quicker model generation and updates. This efficiency gain will lead to cost reductions in terms of manpower and resources.

- Accuracy and Reliability: Automation minimizes the possibility of human errors and inconsistencies, resulting in more accurate and reliable environment models. This, in turn, enhances the safety and performance of autonomous driving systems.

- Scalability and Adaptability: An automated tool chain can be easily scaled to handle larger datasets and accommodate different road traffic scenarios. As the automotive industry advances and expands, the tool chain can be adapted to incorporate new sensors, technologies, and data sources.

4. Limitations

Accuracy and Precision

Achieving high levels of map accuracy is a tough problem due to a variety of reasons, including sensor data quality, the ever-changing nature of road conditions, and the dynamic environment in which autonomous vehicles operate. [6]

One of the biggest limitations on the accuracy of the maps is the data sensor variability. Autonomous vehicles rely on various sensors to collect data about the surroundings, [7] but these sensors can have inherent limitations, such as sensor noise and limited resolution among others.

The biggest limitation on the precision is how the environment can replicate the real-life conditions of the road. Factors like road maintenance and temporary construction can alter the road infrastructure. These conditions can result in navigation errors and complications for autonomous vehicles.

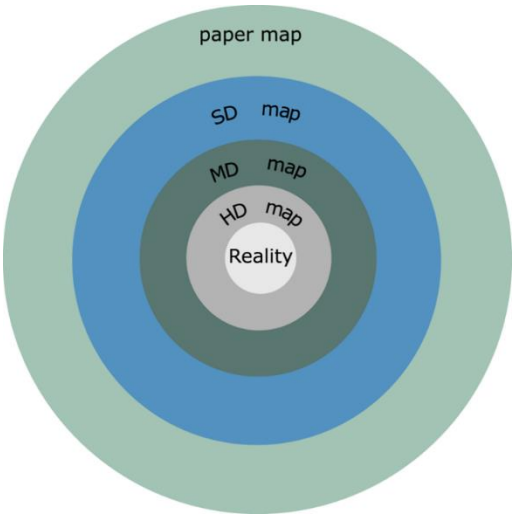

Figure 1: *Representation of the type of maps according to theri accuracy[8]*

As shown in Figure 1, the closest type of map to reality is the HD maps creation and generation are maintenance demanding and expensive. The maintenance of this kind of map is more complicated than the creation because HD map requires quality data that should be supplied quickly and cheaply. [9] A solution for this problem is to use less accurate maps such as MD maps, which are quite accurate for traffic, surrounding elements, and spatial accuracy but they have a reduced computational and economic cost. The MD divides the dense and complex information of HD into more manageable blocks and reduces the requirements for the system. [10]

Data Volume and Storage

The process of generating HD maps involves capturing and storing vast amounts of high-resolution data, the size of this data depends on the area of application, however, a general guideline is required 10MB per km^2 [11]. In the case of the size of the map the guideline is that is required at least 10KB per Km of road. [15][16] This includes information about road geometry, lane marking, traffic signs and other intricate features. This information requires substantial storage capacity and computational resources.

Maintaining large volumes of map data and ensuring real-time updates can incur significant costs. These costs encompass data storage infrastructure, data transfer, and the computational power required for processing and maintaining these datasets.

Localization Challenges

Accurately localizing autonomous vehicles within a digital map can be intricate, especially in maps characterized by tall buildings that obstruct GPS signals or in regions with limited signal reception. Localization errors can have cascading effects on navigation, accentuating the need for precise positioning.

The process of aligning sensor data with digital map data can be challenging, especially when dealing with complex intersections, irregular road geometries, or inaccuracies in map data. [9] Misalignment can result in erroneous localization. Addressing these challenges is critical to achieving precise and reliable vehicle localization within digital maps.

Matching information with world coordinates when generating HD maps involves a process known as georeferencing alignment.[14] This process ensures that the data collected for the map corresponds to its real-world location. However, there are various challenges, such as the georeferencing algorithm, which aligns the sensor data with existing geographic coordinate systems, the reference data sources, which may have inaccuracies because they rely on GPS data.

5. State of the art

The state of the art in developing digital environments for autonomous driving involves the use of advanced technologies such as sensor fusing, computer vision, and 3D modeling.

Sensor fusion is the process of combining data from multiple sensors to create a unified and accurate view of the environment or an object. [15] This can involve different types of sensors such as lidar, radar, cameras, or GPS. The challenges of sensor fusion include dealing with the heterogeneity and complexity of sensor data, which may have different formats, resolutions, sampling rates, coordinate systems and error models.

Computer vision algorithms can then analyze this data to identify and track objects such as vehicles, pedestrians, and road signs. Some of the computer vision algorithms used in Autonomous Vehicules are the object detection algorithms, semantic segmentation, or instance segmentation. Object detection algorithms are essential for identifying and locating objects within the vehicle's environment. [16] The semantic segmentation ones assign a class to each pixel in an image, allowing the vehicle to understand the structure of the surrounding environment. Lastly instance segmentation not only categorizes pixels into classes but also differentiates between individual objects within the same class. [17]

SLAM techniques are employed to create environment models in real-time while simultaneously estimating the position and orientation of the sensor or vehicle. By fusing sensor data, such as LiDAR, cameras, and inertial measurement units (IMUs), SLAM algorithms reconstruct the environment and create a map.

Cloud computing and distributed processing have emerged as powerful tools for environment model creation.[18] By leveraging the computational power and scalability of cloud infrastructure, complex processing tasks, such as large-scale point cloud processing and deep learning-based analysis, can be efficiently performed. Cloud-based solutions provide flexibility, accessibility, and cost-effectiveness for environment model generation.

Overall, the state of the art in developing digital environments for autonomous driving involves the integration of multiple advanced technologies and techniques, with the goal of creating highly accurate and reliable models of the environment that can be used for testing and validation of autonomous driving systems.

5.1. Data Collection

LiDAR

LiDAR (Light Detection and Ranging) is a commonly used technique for environment data collection. LiDAR sensors emit laser beams and measure the time it takes for the laser to return after hitting an object. This data is used to generate precise three-dimensional point clouds, capturing detailed information about the environment's geometry and structure.

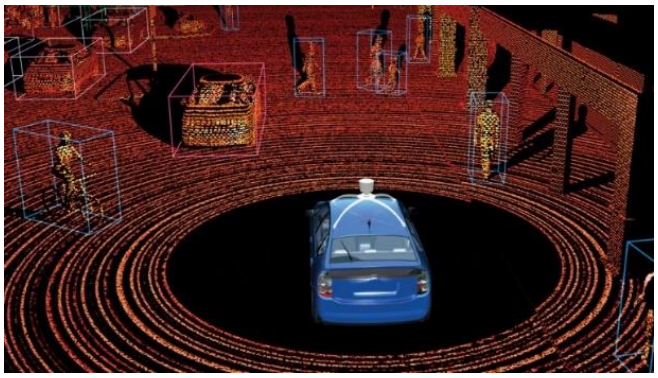

Figure 2: Representation of the measurement with LiDAR sensors

A new kind of Lidar used in the automotive industry are the FMCW (Frequency Modulated Continuous Wave) or 4D Lidar is a type of sensor that measures distance using optical interference frequency.[19] It operates by continuously transmitting a signal with a modulated frequency. When the signal reflects off an object, the Lidar detects the reflected signal and compares it to the original, allowing it to measure the distance and velocity of the object.

But however, it's important to note that while FMCW Lidar has many advantages, it also faces some challenges such as the cost of the components or the lateral movements recognition.

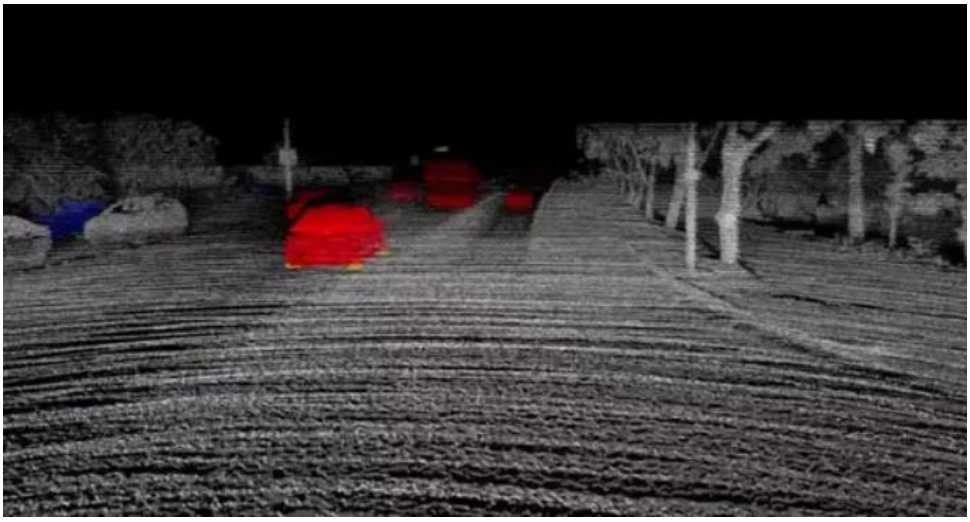

Figure 3:Aeva's FMCW LiDAR that can estimate velocities and predict trajectories (blue: approaching | red: receding)

Photogrammetry and Image-Based Data Collection

Photogrammetry and image-based data collection are essential techniques for understanding the environment in autonomous driving. Photogrammetry uses photos to create precise 3D models, while image-based data collection involves cameras capturing real-time visual data. [20]

Photogrammetry analyzes overlapping images to measure distances, object positions, and textures as shown in Figure 4. It excels at detailed terrain mapping and urban maps, lane recognition, and making it valuable for creating complex digital environments.

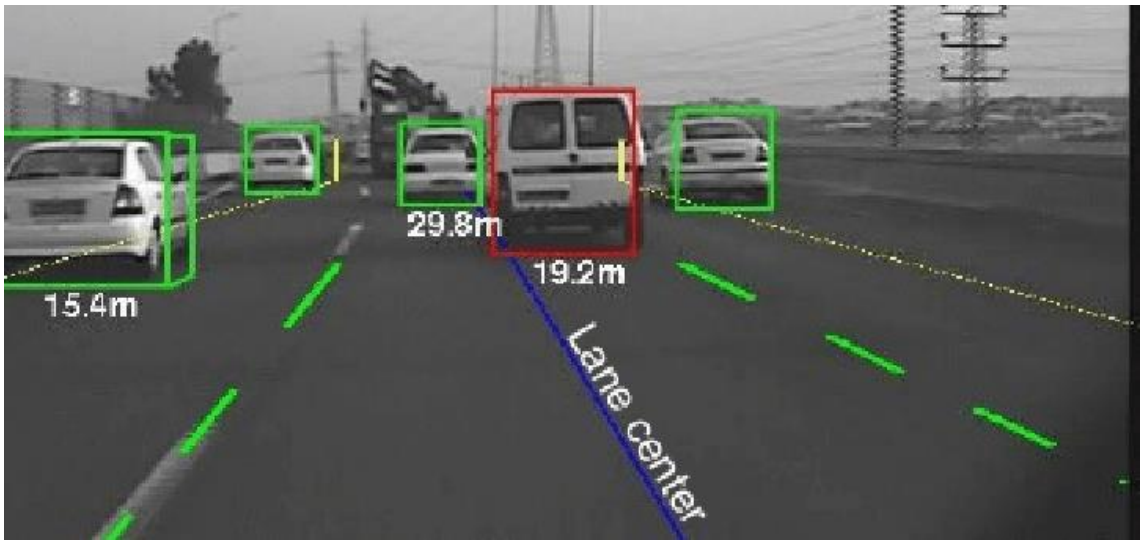

Figure 4: Sample of the real-time artificial vision camera with detected objects for autonomous driving using Mobileye

GPS and IMU Data Collection

GPS technology serves as a foundational data source for digital environment generation in autonomous driving applications. It relies on a network of satellites orbiting the Earth, with GPS receivers on autonomous vehicles precisely calculating their geographic position. These receivers' triangulate signals from multiple satellites to determine latitude, longitude, and altitude coordinates, forming the basis for mapping the vehicle's location within the environment. [21]

While GPS offers extensive coverage and global positioning capabilities, its accuracy and reliability can be influenced by various factors. In urban maps with tall buildings or dense foliage, signal obstructions can lead to inaccuracies, commonly known as the GPS multipath effect. [22] To address these limitations

and enhance accuracy, autonomous vehicles often incorporate multiple sensors, including IMUs and odometry, [23] into their data collection systems. This sensor fusion approach helps correct GPS inaccuracies and ensures a more reliable digital environment.

IMUs excel at capturing rapid changes in vehicle motion, making them essential for tasks like tracking vehicle dynamics, detecting abrupt maneuvers, and maintaining accurate positioning when GPS signals are temporarily lost. These units consist of accelerometers and gyroscopes, providing continuous measurements of an autonomous vehicle's acceleration and angular velocity. [24] For example, when an autonomous vehicle enters a tunnel or navigates through urban environments with tall buildings, GPS signals may become temporarily unreliable. IMUs step in to bridge these gaps in data, ensuring uninterrupted vehicle tracking and localization.

However, IMUs are not without challenges. Over time, IMUs may experience sensor drift, which can lead to positioning errors if not carefully calibrated and compensated for. The time of experience this phenomenon depends on the system developed, but for example in the case of the AUTOPIA Automatic Driving systems they became a good response just on the first 5 minutes of drive test, which is unacceptable for a commercial solution. [25]

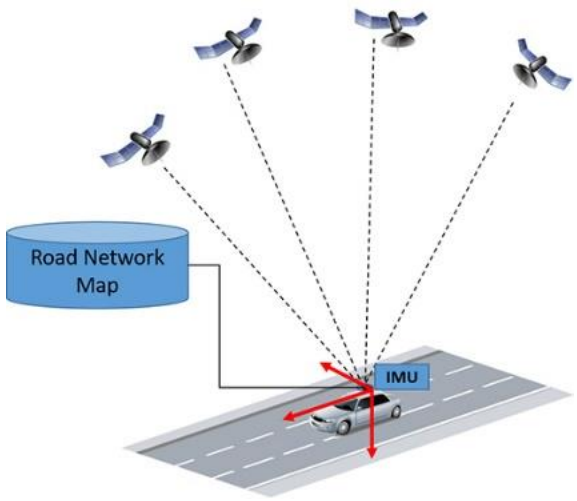

***Figure 5**: Graphic representation of the GPS and IMU*

In summary, GPS and IMU technologies play vital roles in data collection for digital environment generation in autonomous driving. GPS provides global positioning data, while IMUs offer real-time information about vehicle movement and orientation. The synergy between these technologies, complemented by careful calibration and sensor fusion, ensures the creation of highly accurate and reliable digital maps. [26]

Radar Data Collection

Radar technology is a significant contributor to data collection for digital environment generation in autonomous driving. Radar systems emit radio waves or microwave signals and measure the time it takes for these signals to bounce off objects and return to the sensor.

Figure 6 Representation of the radar sensor

Radar's strength lies in its ability to operate effectively in various weather conditions, including rain, fog, and snow, as radio waves are less affected by adverse weather than optical sensors. [27]

This type of sensors can detect objects at longer ranges than many other sensors, making them particularly useful for detecting vehicles, pedestrians, and obstacles at highway speeds. The technology comes in various forms, including

short-range radar for parking and blind spot detection, mid-range radar for adaptive cruise control, and long-range radar for highway and urban scenarios. [28]

However, radar technology does have limitations, such as the limited angular resolution, being its azimuth angle is ± 75 ° horizontal and 12 ° vertical [29], and the difficulty in distinguishing between closely spaced objects. To address these challenges, advanced signal processing techniques and sensor fusion with other sensors like cameras and LiDAR are employed to create a more comprehensive digital environment.[29]

Other solution is the use of 4D Radars, this solution is based on the concept of MIMO(Multiple Input Multiple Output) antennas. [19] For this type of technology dozens of mini-antennas are sending waves all over the place, both horizontally and vertically directions. In a normal 3D RADAR, it's only done horizontally, this type of technology lacks the height, and also has a pretty bad resolution. This new technology also opens the door to detect obstacles and classify them

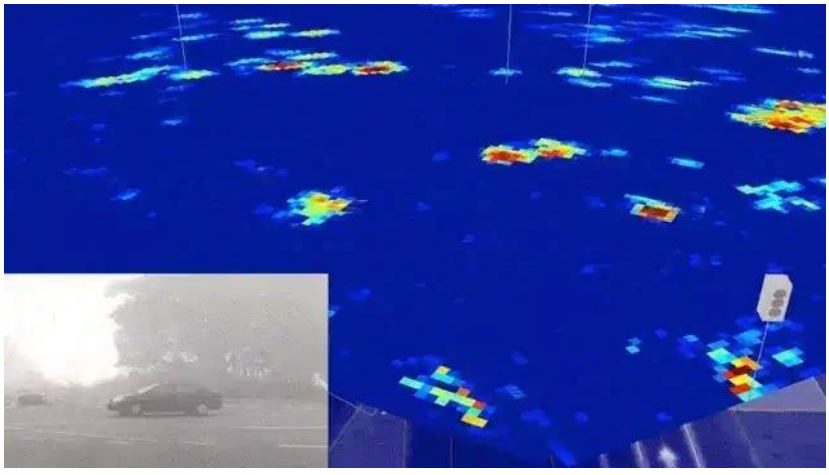

Figure 7: Waymo's Imaging RADAR [30]

In conclusion, radar technology plays an important role in data collection for the construction of digital maps in the context of autonomous driving. It excels at

object detection, distance, and speed measurement, and works well in inclement weather. [22] Integration with other sensor data improves the digital environment perception system's comprehensiveness and reliability, which is critical for enabling safe and effective autonomous driving.

Mobile Mapping Systems

MMS are typically mounted on vehicles and equipped with a combination of advanced sensors, including LiDAR, cameras, GNSS, and IMUs. [8]

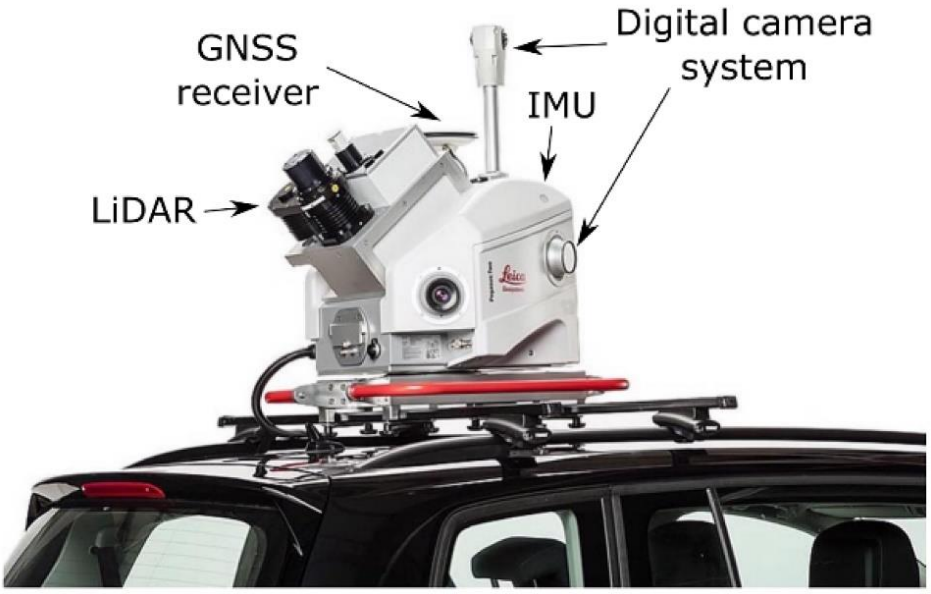

Figure 8:Example of an MMS: a vehicle-mounted mobile mapping platform consisting of different positioning and data collection sensors to generate an accurate georeferenced 3D map of the environment. [23]

Mobile Mapping Systems are adept at capturing dynamic information about the environment. As vehicles equipped with MMS sensors traverse different areas, they continually update the digital map, allowing for real-time tracking of changes in the environment, such as road conditions, construction activities, or new infrastructure developments.[24]

While MMS offers numerous advantages, it is essential to address challenges such as data volume management and processing complexity. The sheer volume of data generated by MMS sensors can be substantial, necessitating robust data storage and efficient processing pipelines to create and update digital maps effectively.[23]

Sensor fusion

Sensor fusion is a technique commonly used in autonomous drive systems. This technique merges the data from multiple sensors to build more accurate , reliable, and robust world model for the car to navigate and behave more successfully.

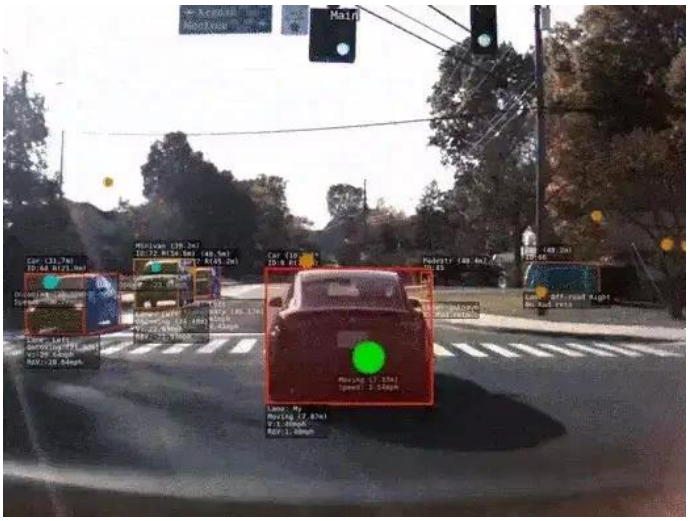

Figure 9: Radar fused with camera

For the fusion of the sensor outcomes the most common type of algorithm for the fusion is by abstraction level. Inside the abstraction level in the industry are three different processes: Low-Level, Mid-Level, High-Level.

Low Level Sensor Fusion is about fusing the raw data coming for multiple sensors. For example, fusing point clouds coming from Lidar and pixels coming from cameras. This type of fusion was very hard to do until a few years ago, because the processing required is huge because at each millisecond, is possible to fuse hundreds of thousands of points with hundreds of thousands of pixels.

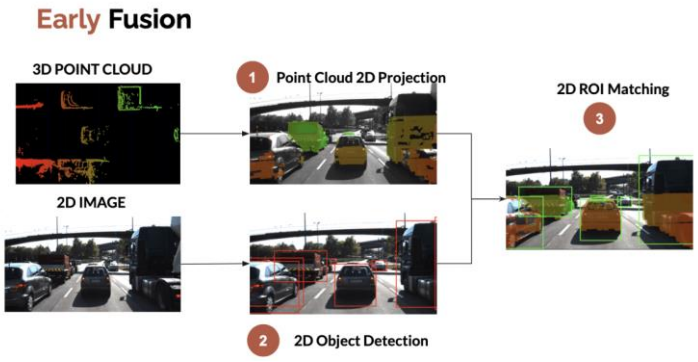

Figure 10: Simplified diagram of early fusion technique

Early Fusion combines data, in the example Lidar point cloud and camera images, before processing it. It involves transforming Lidar data to 2D images applying deep learning as R-CNN or YOLO for object detection. [31] The Lidar sensor utilizes object detection by projecting 3D point clouds into 2D images and associating them with pixels. This method facilitates the algorithm's ability to identify objects within its field of view.

Mid-level fusion enhances object detection by integrating independently detected objects from various sensors. For instance, if both a camera and radar detect an obstacle, their findings are fused to provide the most accurate estimation of the obstacle's position, class, and velocity. While the Kalman Filter is a common approach for such fusion, its reliance on sensor data poses a drawback; failure of any one sensor can compromise the entire fusion process.

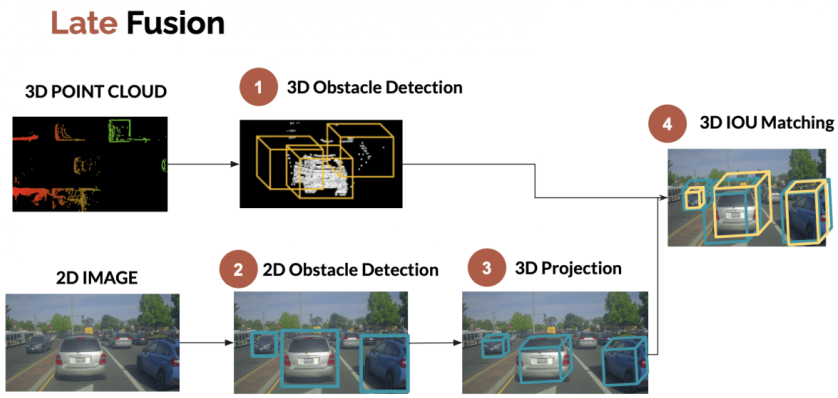

Figure 11: Simplified diagram of late fusion technique

In the Late Fusion approach Lidar and camera data are processed independently. This involves detecting and localizing objects in the 3D using Lidar data, while simultaneously conducting object detection in the 2D with camera data. Subsequently, the detected objects from the camera are projected onto 3D to align them with Lidar-detected objects. Finally, a union matching process is employed to merge the results from both sensors into a comprehensive and accurate final detection result.

AV Application	*Fused Sensors*	*Limitation without Fusion*	*Advantages with Fusion*
Object Detection	Lidar & Camera	Night vision, illumination, low Lidar resolution	Improved depth, extended range, and enhanced accuracy, robust perception in varied conditions
Localization & Mapping	GPS and Lidar	Poor GPS in denied areas	Continuous navigation, precise localization, enhanced mapping in diverse environments
Positioning & Navigation	Lidar Map, Camera and GPS	GPS-denied areas, road marking limitations	Accurate road marking detection, integration with HD maps, robust navigation capabilities
Perception in Bad Weather	Lidar, Camera, and Radar	Limited performance in bad weather (fog, rain)	All-weather solution for AVs, reliable operation in adverse weather conditions

Table 1 Comparison of different sensor fusion types

This table provides an overview of various data collection methods, highlighting their respective accuracies measurement times, distance of measurement and equipment costs.

Data Collection Method	*Accuracy (cm)*	*Measurement time (ms)*	*Distance of Measurement (m)*	*Equipment Cost (€)*
LiDAR	High (1-10)	Fast (100-1000)	0.2-150	10,000 - 100,000
Camara Based	Moderate (10-50)	Fast (10-100)	Up to 100	1,000 - 10,000
GPS and IMU	Moderate (100)	Continuous	N/A	100 - 1,000
Radar	Moderate (20-50)	Fast (10-100)	Up to 200m	5,000 - 50,000
Mobile Mapping Systems	High (10-50)	Continuous	Up to 150m	50,000 - 500,000

Table 2 Comparison of the different data collection methods

In conclusion, in terms of performance the best solution for the data collection is the mobile mapping system, which is the combination of some of the previous methods. On the other hand, GPS and IMU or Camara based methods are a more affordable solution.

5.2. Map file format

Open source

There are several open-source mapping options available for creating digital maps. However, based on popularity and qualities, OSM and OpenDRIVE are the best options. Both file formats are used to represent road networks in a digital format, but they have some key differences.

OpenStreetMap (OSM)

OSM is a collaborative, open-source mapping platform that provides free and editable geospatial data. This platform offers a large and detailed dataset of road networks and related infrastructure. OSM data offers extensive information about road layout, traffic laws, lane configurations, traffic signs, junctions, and other critical features for effectively simulating road maps.

The files are created and maintained by a community of volunteers who use GPS devices and other tools to collect information about roads and other features.[32]

Structure of the data

The data structure of an OSM file is based on a specific XML format designed to store geospatial data. [33] OSM files contain information about various geographic features, such as roads, buildings, points of interest, and more. The structure of an OSM file can be categorized into three main components: nodes, ways, and relations, as is shown in Figure 6. [34]

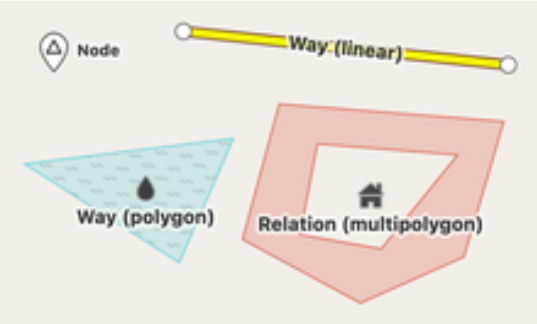

***Figure 12**: Representation of the different components of an OSM file [35]*

Nodes

A node element represents a single point in space with a unique identifier and is defined by its latitude and longitude coordinates, as shown in Figure 13.

```
<node id="25496583" lat="51.5173639" lon="-0.140043" version="1" changeset="203496" user="80n" uid="1238" visible="true"
timestamp="2007-01-28T11:40:26Z">
    <tag k="highway" v="traffic_signals"/>
</node>
```

Figure 13: *Example of a node[36]*

In an OSM file, nodes are typically used to represent specific locations or points of interest, such as intersections, landmarks, or individual features like traffic signs or trees.

Nodes may also have additional tags associated with them to provide further descriptive information. [36] As an example of additional tags is the use of ele=* tag and its subkeys to give the information of the elevation to the node. [1][37]

The structure of a node in an OpenStreetMap (OSM) file consists of several key components that define its properties and attributes.[36] Each node represents a specific point in space, identified by a unique identifier, as shown in Figure 8. The structure of a node is as follows:

Node ID: Nodes have distinctive identifiers called node ids. Node ids on the server are durable, therefore no matter how many times data are updated or rectified, the allocated id of an existing node will not change. Unless a previous node has been undeleted, deleted node ids must not be used again.

Latitude: Latitude coordinate in degrees (North of equator is positive) using the standard WGS84 projection.

Longitude: Longitude coordinate in degrees (East of Greenwich is positive) using the standard WGS84 projection.

Tags: Tags are key-value pairs that provide additional descriptive information about the node allowing to label and categorize nodes for more specific and meaningful representation of geographic features.

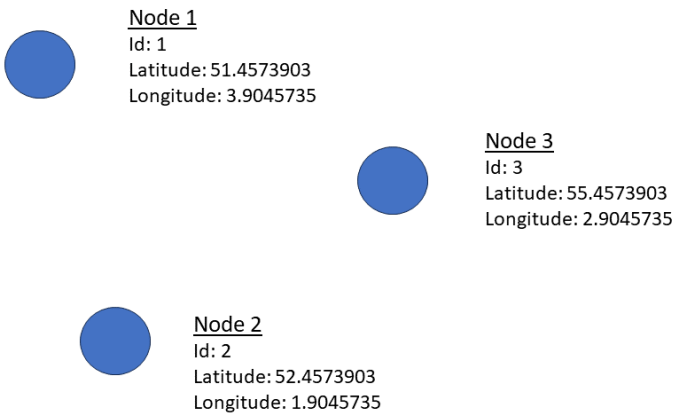

Figure 14: *Representation of nodes with different coordinates*

name	value
Id	64-bit integer number (≥ 1)
lat	decimal number ≥ −90.0000000 and ≤ 90.0000000 with 7 decimal places
lon	decimal number ≥ −180.0000000 and ≤ 180.0000000 with 7 decimal places
tags	A set of key/value pairs, with unique key

Table 3: *Representation of the values for a node in OSM [36]*

Ways

A way represents a collection of nodes that form a linear feature, such as a road, path, or boundary. Ways are defined by an ordered list of node references, where each reference corresponds to a node identifier, as sown in Figure 9. [38]

```
<way id="5090250" visible="true" timestamp="2009-01-19T19:07:25Z" version="8" changeset="816806" user="Blumpsy" uid="64226">
  <nd ref="822403"/>
  <nd ref="21533912"/>
  <nd ref="821601"/>
  <nd ref="21533910"/>
  <nd ref="135791608"/>
  <nd ref="333725784"/>
  <nd ref="333725781"/>
  <nd ref="333725774"/>
  <nd ref="333725776"/>
  <nd ref="823771"/>
  <tag k="highway" v="residential"/>
  <tag k="name" v="Clipstone Street"/>
  <tag k="oneway" v="yes"/>
</way>
```

Figure 15: *Example of a way [38]*

Ways can also have various tags associated with them to provide additional information, such as the type of road, speed limits, or lane configurations. The structure of a way is as follows:

Way ID: Each way is given a distinct identification known as the way ID, much like nodes are. This ID is used for linking and referencing reasons as well as to distinguish the route from other dataset parts.

Node References: A way is defined by an ordered list of node references. Each node reference corresponds to the unique identifier of a node that is part of the way. These node references define the sequence and connectivity of nodes along the path of the way, thus creating the linear feature. The order of node references determines the spatial geometry of the way.

Tags: Ways can have tags associated with them to provide additional descriptive information. Tags are key-value pairs that describe various attributes of the way, such as its type, name, surface condition, speed limit, or any other relevant characteristics. Tags help classify and categorize the way based on its attributes.

Relation

Relations enable the grouping of numerous nodes, ways, or other relations based on a shared relationship or theme association. Relations are defined by a set of members, each of which is identifiable by its type (node, way, or relation) and matching identifier, as shown in Figure 10.[39]

```
<relation id="13092746" visible="true" version="7" changeset="118825758" timestamp="2022-03-23T15:05:48Z" user="" uid="">
 <member type="node" ref="5690770815" role="stop"/>
 <member type="node" ref="5751940550" role="stop"/>
 ...
 <member type="node" ref="1764649495" role="stop"/>
 <member type="way" ref="96562914" role=""/>
 ...
 <member type="way" ref="928474550" role=""/>
 <tag k="from" v="Encre"/>
 <tag k="name" v="9-Montagnes de Guyane"/>
 <tag k="network" v="Agglo'bus"/>
 <tag k="not:network:wikidata" v="Q3537943"/>
 <tag k="operator" v="CACL"/>
 <tag k="ref" v="9"/>
 <tag k="route" v="bus"/>
 <tag k="source" v="https://www.cacl-guyane.fr/wp-content/uploads/2021/01/PLAN-RESEAU-URBAIN-AGGLO-BUS-1.pdf"/>
 <tag k="to" v="Lycée Balata"/>
 <tag k="type" v="route"/>
 <tag k="website" v="https://www.cacl-guyane.fr/lagglo-au-quotidien/se-deplacer/transport-urbain-2/"/>
</relation>
```

Figure 16: *Example of a relation[39]*

Relations are frequently employed to represent complicated characteristics or geographical connections such as multi-polygon borders, route relations, or administrative boundaries. The structure of a way is as follows:

Relation ID: Each relation in an OSM file is assigned a unique identifier known as the relation ID. This ID is used for referencing and linking purposes.

Members: A relation is defined by a collection of members, where each member is identified by its type (node, way, or relation) and its corresponding identifier. Members can be nodes, ways, or even other relations. The members define the elements that are part of the relation and establish the relationship between them. The order of members within a relation is not significant.

Roles: A role describes the specific function or purpose that the member plays within the context of the relation. Each member in a relation can have an associated role. For example, a member may have a role of "inner" or "outer" in a relation representing a multi-polygon boundary.

Tags: Relations can also have tags associated with them to provide additional descriptive information. Tags are key-value pairs that describe attributes of the relation, rather than the individual members. Tags can provide information such as the type of relation, a name, or any other relevant attributes.

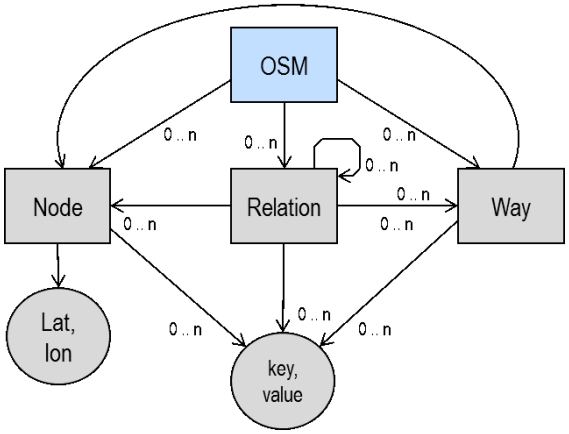

Figure 17: *Structure of the OSM file[35]*

Summarizing OSM structure, this type of map consists of four main elements such as nodes, ways, and relations, each represented by XML tags within the file as represented in Figure 17.

Open DRIVE

Open DRIVE is a standardized file format used to represent road networks and related information. This format is specifically designed to represent road

networks for use in autonomous driving simulations and other advanced driving applications including information about traffic signs and signals, and other infrastructure features that are important for autonomous driving systems. [40]

Structure of the data

The structure of an Open Drive file follows a hierarchical organization, consisting of different levels that represent various aspects of road networks, as shown in Figure 12.

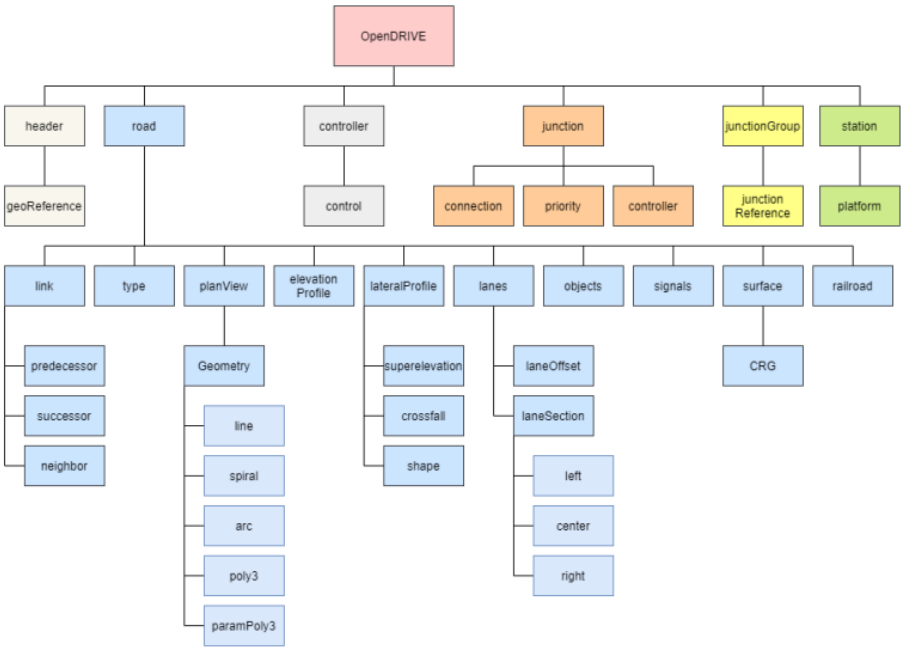

Figure 18: *Structure of OpenDrive file [15]*

The structure can be categorized into six main aspects: road network level, road level, lane level, object level and junction level.

Road Network Level: The road Network Level, which is the highest level, provides the overarching framework for representing the entire road system. It includes the overall characteristics and metadata of the road network, such as the reference

coordinate system, measurement units, and background data. This level guarantees uniformity and consistency over the whole network representation.

Road Level: The road level focuses on individual road segments within the network. Each road segment represents a specific stretch of roadway with its unique identifier, name, length, and road type. This level captures the geometric attributes of the road, including the centerline geometry, lane widths, lane count, and information about the road's physical characteristics such as curvature and elevation profile. It provides the foundation for accurately representing the shape and layout of each road segment.

Lane level: The lane level goes into the qualities and attributes of specific lanes within a road section. It defines lanes as separate entities, each one with its own identifier, width, and attributes. Information at the lane level includes elements like lane markings, lane limits, and lane kinds (such as driving, turning, or shoulder lanes). This level allows the accurate representation of lane configurations, ensuring the precise modeling of road infrastructure and supporting lane-specific behavior in autonomous driving simulations.

Lane Section Level: The lane section level further refines the representation of lanes by dividing them into sections along the road segment. It allows for the modeling of variations in lane properties such as changes in width, slope, or lane markings within a lane segment. By defining lane sections, the Open DRIVE format accommodates accurate representation of lane characteristics and facilitates the modeling of complex road geometries.

Object Level: The object level incorporates additional elements associated with the road network, such as signs, traffic lights, barriers, or other objects of interest. Objects are defined with their respective positions, dimensions, and attributes. This level enables the placement of objects along the road segment, providing contextual information about the road environment that influences autonomous

driving behavior. Objects can represent both static features and dynamic entities like moving vehicles, pedestrians, or other dynamic traffic elements.

Junction Level: The junction level captures the complex interactions and connections between different roads within the road network. It focuses on modeling intersections, interchanges, roundabouts, or any other form of road junctions. Junctions are defined with their geometry, traffic rules, and connections between incoming and outgoing roads. This level provides a detailed representation of the geometric layout, lane connectivity, traffic priorities, and regulatory information necessary for accurate simulation and analysis of complex road scenarios.

	OSM	Open DRIVE
Data Source and Level of Detail	Crowdsourced, varying detail and accuracy	Standardized, detailed road information
Standardization and Industry Compatibility	Lack of strict standardization, flexible	Standardized format, better integration
Integration with Simulation Environments	Requires additional processing for simulation use	Designed for direct integration with simulations
Lane Geometry and Traffic Rules Representation	Limited representation, may require additional work	Detailed lane geometry and traffic rule information
Compatibility with Autonomous Driving Systems	Requires data preprocessing and conversion	Direct compatibility with autonomous systems
Support for Complex Road Features	Inconsistent support for complex features	Comprehensive support for complex road attributes
Elevation Data	Available but limited in detail and coverage	Comprehensive elevation data for road modeling

Table 4: *Comparison of OSM and OpenDRIVE format [2], [35], [36], [38], [39]*

In summary, OSM files are used to represent maps and routing information for general-purpose applications, while OpenDRIVE files are designed specifically for advanced driving simulations and autonomous driving applications. OpenDRIVE files contain much more detailed information about the road

network, making them more suitable for use in applications that require precise information about the road environment.

Commercial solutions

It is more difficult to determine the technical details of commercial solutions because they usually reserve the information for customers only, however it is true that mapping businesses are also working on HD map solutions.

NVIDIA DRIVE

NVIDIA DRIVE Mapping module is part of the NVIDIA End-to-End Autopilot Systems solution. [41] It's a scalable system that incorporates a sensor suite, software, and software APIs, as well as HD maps from mapping businesses. It comprises of the following components:

- DRIVE Localization, which determines the precise 6-DOF location and orientation of an autonomous vehicle inside an HD map with centimeter-level precision.
- Drive Map stream for updating cloud-based HD maps with DRIVE Perception Road characteristics.

HERE

HERE HD Live Map is part of the solution provided by Here Technologies. It utilizes machine learning algorithms to validate map data in real-time, ensuring its alignment with real-world conditions. [42] By analyzing a multitude of data

sources, such as satellite imagery and sensor data, the self-healing map system continuously updates and refines the map accuracy.

Google Maps API

The Google Maps API is the solution provided by Google. It provides a robust tool for enhancing High-Definition (HD) map generation in autonomous driving applications. While not explicitly designed for this purpose, the API offers a rich array of features that significantly contribute to the development and deployment of HD maps for autonomous vehicles.

By providing access to extensive map data, including road networks and landmarks, it serves as a foundational layer for detailed HD map creation, encompassing vital information like road layouts, lane markings, and traffic signs. Additionally, integration with Google Street View allows for the incorporation of panoramic imagery, providing real-world visual context that aids in precise navigation and object recognition, ultimately optimizing the effectiveness of autonomous vehicle systems. [43]

	Open-Source Solutions	Commercial Solutions
Data Source and Level of Detail	Crowdsourced, varying detail and accuracy	High level of detail and accuracy
Standardization and Industry Compatibility	Lack of strict standardization, flexible	Industry-standard formats and protocols
Real-Time Updates and Currency	Near real-time, potential delays	Real-time updates and data streaming
Integration with Simulation Environments	Requires additional processing for simulation use	Designed for direct integration with simulations
Compatibility with Autonomous Driving Systems	Requires data preprocessing and conversion	Optimized for autonomous driving applications
Support for Complex Road Features	Inconsistent support for complex features	Detailed elevation data for accurate road representation
Elevation Data	Available but limited in detail and coverage	Detailed elevation data for accurate road representation
Cost	Free	Licensing costs

Table 5: *Comparison between Open-Source and commercial solutions*

5.3. Elevation data

Open-Source

NASADEM

NASADEM is a high-precision digital elevation model dataset developed by NASA's Earth Science Division. Although is not explicitly designed for HD map generation for autonomous driving as in the case of Google Maps API, NASADEM provides very interesting elevation data that can be used in enhancing the accuracy and detail of them.

The dataset offers precise elevation data for the Earth's surface, derived from a combination of remote sensing instruments, including the Shuttle Radar Topography Mission (SRTM) and the Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER).[44]

NASADEM's elevation data is essential for understanding environmental factors like slope, aspect, and elevation zones. This information can be used in conjunction with other data sources in HD maps to support eco-friendly driving strategies, optimizing vehicle performance based on the terrain and environmental conditions.

OpenTopography

OpenTopography is an open source tool developed by the University of California San Diego to advance our understanding of the Earth’s surface, vegetation, and built environment [45]. Again, the tool is not designed explicitly for autonomous driving, but this technology offers valuable resources that can enhance the precision and detail of HD maps, making it an asset for research and development in this field.[46]

OpenTopography serves as a vast repository of high-resolution elevation data, primarily derived from airborne LiDAR (Light Detection and Ranging) surveys. These surveys employ laser technology to capture precise elevation measurements, resulting in incredibly accurate and detailed topographic data. For HD map generation, access to such high-quality elevation data is fundamental.[47] As in the case of NASADEM data, the primary application of OpenTopography's data is terrain modeling.

Although developing maps for the European market poses a challenge due to the limited availability of topographic maps compared to the U.S. Unlike the extensive data for the U.S., European regions lack equivalent topographic offerings. This scarcity necessitates a nuanced approach, possibly involving alternative geospatial datasets.

Commercial Solutions

Google Maps Elevation API

The Google Maps Elevation API is a versatile tool developed by Google. This tool can enhance the precision and detail of High-Definition (HD) maps, particularly in the context of elevation data and terrain modeling. As in the case of map generation this tool was not originally designed for autonomous driving, but it offers valuable features and data access that can significantly benefit map developers and researchers in this field.

The API offers the potential for real-time elevation data retrieval, which can be beneficial for dynamic map updates. This feature is particularly useful in scenarios where elevation changes occur due to construction, road closures, or natural events.

Google Maps Elevation API can be seamlessly integrated into various mapping and GIS (Geographic Information System) platforms, streamlining the workflow for HD map generation. This integration simplifies the process of incorporating elevation data into the map creation pipeline.

In addition Google’s cloud infrastructure also supports the API, facilitating data storage, processing, and accessibility. This cloud-based approach can be advantageous for HD map generation, especially when managing and analyzing large datasets.

Solution	Data Source	Aviability	Coverage Area	Resolution	Cost
NASADEM	SRTM, ASTER GDEM	Open-Source	Global	30-90 m [3]	Free
OpenTopogrphy	Airborne LiDAR, Terrestrial LiDAR	Open-Source	Various regions (Mainly US)	0.5m [48]-90m [49]	Free
Google Maps Elevation API	DEM SRTM	Commercial	Global	0.3-9m [50]	0.05$/Requests[51]

Table 6: *Comparison of the Elevation data Solutions [1] [2] [3]*

5.4. Software tools

There are several software tools and platforms available that can simplify the process of modeling streets for autonomous driving. The most developed projects are CARLA, SUMO, and RoadRunner.

Open-Source

CARLA

CARLA is an open-source simulation platform designed primarily for the development and testing of autonomous driving systems by the CVC in Barcelona. While CARLA's primary focus is on simulating vehicle behavior and interactions, it also contributes to the generation of digital maps crucial for training, testing, and validating autonomous driving algorithms. [52]

CARLA provides a highly realistic 3D simulation environment, replicating real-world road networks, urban landscapes, and diverse terrains. This rich and immersive environment serves as the canvas for digital map generation.

The technology also simulates a variety of sensors commonly used in autonomous vehicles, including LiDAR, cameras, radar, and GPS. These sensors capture data within the virtual environment, which is subsequently used to generate digital maps. Simulated sensors mirror their real-world counterparts, enabling accurate data collection.

CARLA's dynamic weather and lighting system allows for the creation of digital environments with varying conditions, such as different times of day, weather patterns, and lighting scenarios. This versatility enables the simulation of environments under diverse circumstances, essential for comprehensive HD map generation.

It also enables users to annotate maps within the simulation environment. Developers and researchers can add road markings, traffic signs, traffic lights, and other essential features directly onto the digital environment. Unfortunately, the tool does not annotate the elevation.

The software incorporates a realistic traffic simulation system, complete with AI-controlled vehicles and pedestrians. This dynamic traffic environment allows for the generation of maps that accurately represent complex traffic interactions and scenarios. Users can create custom scenarios within CARLA, including challenging driving situations, intersections, and urban maps. These scenarios can be utilized to simulate and generate digital maps with specific attributes and challenges.

SUMO

SUMO is an open-source, highly versatile traffic simulation platform primarily designed for modeling and analyzing urban mobility scenarios developed by the

Institute of Transportation Systems. While its primary purpose is traffic simulation, SUMO indirectly contributes to the generation of digital environments that are essential for autonomous driving research, development, and testing. [53]

SUMO excels in simulating realistic traffic behavior, including vehicle movements, interactions, and traffic flow within urban environments. This capability provides a foundation for creating digital environments that closely mimic real-world traffic conditions. This technology allows users to define intricate road networks, complete with various road types, lanes, intersections, and traffic control measures. These road networks serve as the basis for digital environment generation, offering a high level of customization and realism.

The technology provides functionality for exporting simulation data, including vehicle trajectories, traffic signals, and road network information. This exported data serves as valuable input for HD map generation and digital environment modeling.

SUMO can be integrated with various tools and software, including geographic information systems (GIS) and traffic modeling software as CARLA software.

Users can define custom traffic scenarios and scenarios related to urban mobility within SUMO. These scenarios can range from typical urban traffic conditions to complex intersection interactions, providing a diverse range of digital environments for testing and research.

SUMO offers visualization tools that allow users to view and analyze simulated traffic and digital environments. This visualization aids in understanding traffic dynamics and assessing the realism of the generated environments.

Commercial Solutions

RoadRunner

RoadRunner is a specialized software tool developed by Mathworks designed to work within the MATLAB environment, primarily focusing on road network generation and modeling. [54]

RoadRunner within MATLAB provides the capability to generate complex road networks, including road geometries, lanes, intersections, and road segments. These road networks serve as the fundamental structure for creating digital maps. The technology also incorporates traffic flow modeling algorithms that simulate realistic traffic behavior. This modeling includes vehicle movements, interactions, and the flow of traffic within the generated road networks. The realism of traffic flow contributes to the authenticity of the digital environment.

RoadRunner facilitates the export of simulation data, including vehicle trajectories, traffic signals, and road network details. This exported data serves as valuable input for HD map generation and digital environment modeling.

The software also provides visualization tools that allow users to view and analyze simulated traffic, road networks, and digital environments. Visualization aids in understanding traffic dynamics and assessing the realism of the generated maps.

6. Approach

For the case study, the information used will be open source. Open-source information refers to data, software, and resources that are freely available for public use, modification, and distribution.

Open-source information provides a wide range of easily available and constantly evolving resources. It enables us to take advantage of currently available open-source software, datasets, and libraries that have been created and maintained by the community. This allows us to concentrate on expanding and adapting these resources to meet the unique requirements of the toolchain because it saves us a substantial amount of time and effort during development.

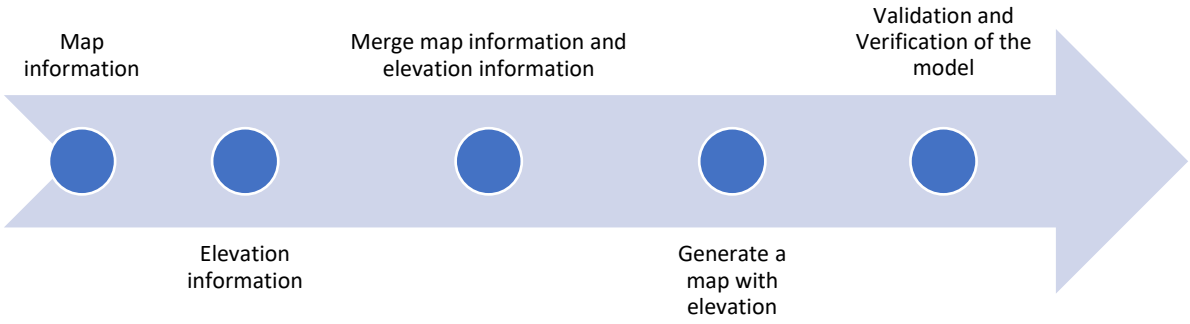

Figure 19 Timeline of the process

6.1. Map information

The choice of OpenStreetMap (OSM) as the main data source for delivering map data to the toolchain was chosen mainly because it was the unique open-source solution available.

The selected area for the map information will be the city of Wolfenbüttel, which is in Lower Saxony, Germany. The geographical coordinates of this area are from 10.5240 E to 10.5678 E for the Longitude and from 52.1742 N to 52.2116 N for the Latitude. This area has been selected due to is a well-known place for us because the university campus is situated in this town.

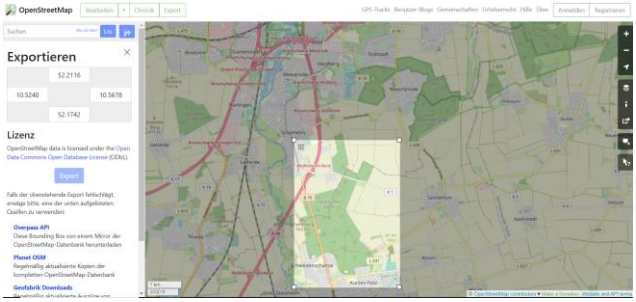

Figure 20: Selection of the studied area in OpenStreetMaps

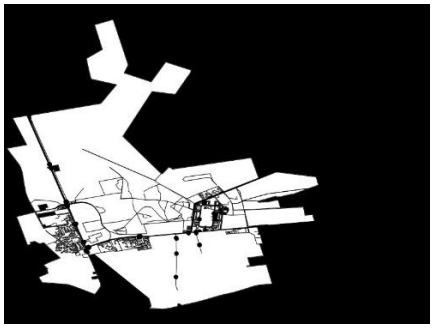

Figure 21: Representation of the OSM map

Before initiating any data treatment, a preliminary representation of the nodes is generated from the designated area to gain insight into the functionality of the OSM data.

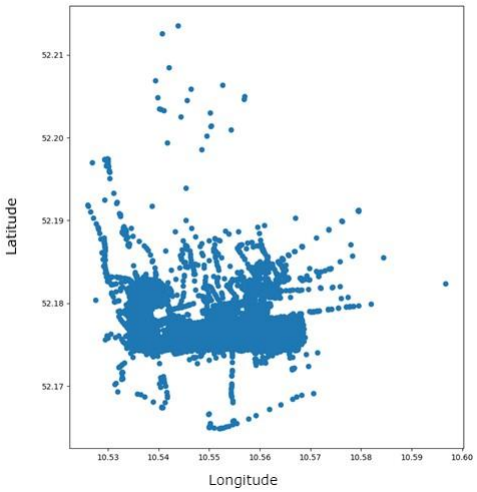

Figure 22: Representation of the map nodes

Through this initial evaluation, a notable observation is made regarding the non-uniform destitution of distances between individual nodes, as illustrated in Figure 22. This inconsistency in distance could stem from the method of data acquisition, typically reliant on GPS sensors. Factors such as fluctuating vehicle speeds may contribute to the irregular spacing between nodes. This discrepancy highlights a potential challenge in accurately capturing and representing spatial data, emphasizing the need for further analysis and potential corrective measures to enhance data quality and reliability.

Initially, the OSM format was considered for the representation of the map data. However, it became evident that this format was not optimal for storing data for autonomous driving applications, primarily due to its node-based representation of environments, which generated an spiky representation in some areas of the roads due to the nodes are connected with straight lines. Following extensive research, the OpenDRIVE format emerged as the most suitable alternative, renowned for its specialized features tailored specifically for autonomous driving needs and its widespread acceptance within the industry.

The OpenDRIVE format offers numerous advantages that position it as the preferred choice for autonomous driving applications, as previously mentioned. However, obtaining OpenDRIVE maps from third-party suppliers proved impossible due to the lack of open-source solutions for this file format. As a result, interest shifted to converters that could convert OSM format files to OpenDRIVE format.

The initial attempt at conversion involved the CARLA simulator converter. [55] While this tool successfully converted the map, it failed to generate the elevation layer, prompting the need to explore alternative solutions. After thorough investigation, osm2xodr emerged as the most suitable tool, meeting the requirement for both map conversion and elevation layer generation.

6.2. Elevation information

The integration of elevation data provides accurate modeling of topography and terrain features in the created digital environments. This allows us for a more precise modeling of elevation variations such as hills or slopes, giving a realistic environments for testing and verifying autonomous driving systems.

In handling elevation data, the osm2xodr tool initially relied on a conversion process to generate the elevation data from a PNG file. However, such conversions are susceptible to errors, potentially resulting in inaccuracies within the elevation map, as seen on Figure 23. To address this issue, a more robust solution is proposed: directly reading the source file and generating the grid from the GeoTIFF format. This approach ensures greater accuracy and reliability in the elevation mapping process.

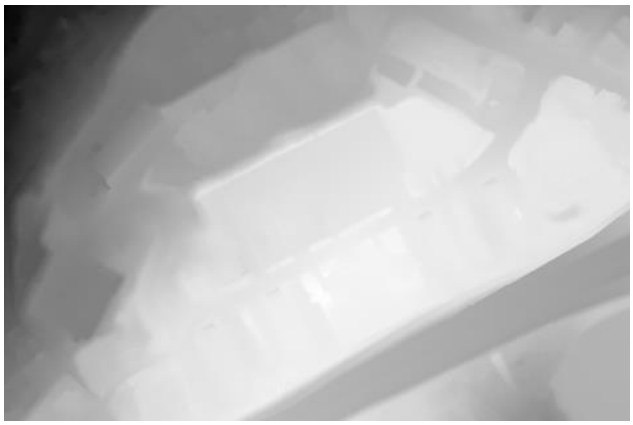

Figure 23: Representation of the elevation data with PNG file

To implement this solution, the tool will utilize the rasterio library for Python. This library provides comprehensive functionality for reading and processing geospatial raster data, offering the possibility to the tool to extract elevation information directly from the GeoTIFF file. By the use of this type of file and accessing the source datan from a GeoTIFF file, the tool can enhance the

integrity and precision of the elevation map generation process, thereby improving the overall quality of the output.

GeoTIFF (Geographic Tagged Image File Format) files were chosen for use in the presented tool due to its inherent capability for handling and storing geospatial data. They may keep georeferencing data and accompanying metadata for multidimensional data like raster pictures. The GeoTIFF format stores raster data by organizing it into a grid of pixels. Each pixel in the grid corresponds to a specific location on the map surface.

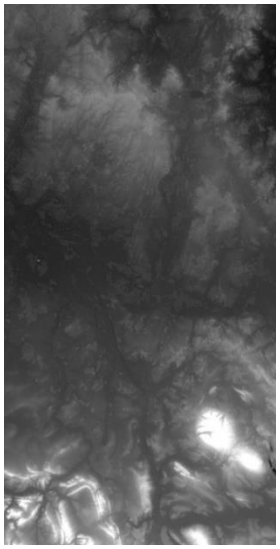

Figure 24: TIFF image of a small area in Niedersachsen

For the case of study, the selected area will cover a larger portion of Lower Saxony due to it being the option which fitted the best to the requirements. The resolution of the selected information was 1arc-second (approximately 30m).

In conclusion, transitioning from PNG conversions to direct GeoTIFF file reading ensures precision using tools like rasterio. GeoTIFF's capability in storing geospatial data provides detailed surface analysis. The chosen study area encompasses a significant portion of Lower Saxony, with a 1 arc-second resolution (approximately 30 meters). Overall, this approach enhances the realism and accuracy of digital environments for autonomous driving system validation.

6.3. Converter

Converting data between different formats is a step needed to the generation of the OpenDrive map. The conversion mainly is focused in two parts: the translation of the map data and the annotation of the elevation to the map file.

The converter is specifically designed to extract detailed information about the road network from the OSM file. This includes essential elements such as roads, lanes, intersections, and other pertinent features necessary for accurate representation. Simultaneously, it retrieves elevation data from the GeoTIFF file, ensuring precise terrain height information is captured and integrated seamlessly into the map.

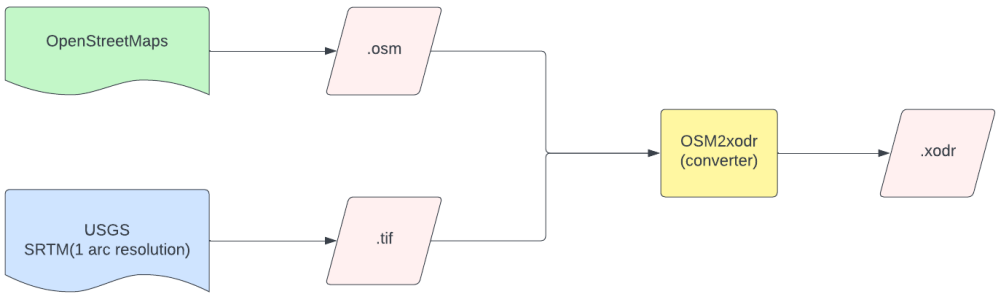

Figure 25: Diagram illustrating the data conversion process for generating OpenDrive maps

Map Data Translation

The successful conversion of map data from OSM to OpenDrive depends on comprehending the profound differences in data structure, semantics, and purpose that distinguish these two formats.

According to chapter 5, in OpenStreetMaps the data is structured around nodes, ways and relations, where nodes represent individual geographical points, ways are ordered lists of nodes that define linear features like roads and relations represent grouping of nodes, ways and other relations.

On the other hand, OpenDrive is a more sprecialized format which focuses on accurately representing road shapes, lane setups, signs, and other essential factors. The core structure of OpenDrive relies on well-defined mathematical equations and geometrical principles. Roads are precisely described using mathematical formulas that capture curves, slopes, and other geometric traits. Lanes are thoughtfully integrated into these equations, allowing us to specify lane widths, positions, and road markings precisely. The way different road segments connect is seamlessly woven together using principles from graph theory.

The conversion approach entails a comprehensive data mapping procedure in which road segments from OSM data are found and turned into drivable lanes inside the OpenDrive schema. This necessitates extracting essential road geometry details, such as lane widths, offsets, and curvature, to preserve the fidelity of the simulation. Moreover, attributes from OSM, such as road names, speed limits, and lane types, must be meticulously mapped to their corresponding counterparts in the OpenDrive format.

Elevation Annotation

Beyond the translation of map data, another critical dimension on the conversion process relate to the annotation of elevation information onto the OpenDrive map. Elevation data plays a pivotal role in generating realistic road profiles and gradients, which are essential for simulating accurate vehicle behavior, such as braking and acceleration responses.

The elevation annotation process involves sourcing accurate elevation data from specialized sources, such as digital elevation models (DEMs).

Understanding the nuanced differences in data structure, semantics and purpose between these formats is crucial for executing a successful conversion. By

precisely translating map data and annotation elevation information, the resulting OpenDrive map creates a solid platform for simulating complicated driving scenarios.

The converter, developed using functional programming techniques in Python, consists of modular components working collaboratively to process, transform, and seamlessly integrate OSM and elevation data into a unified OpenDrive representation.

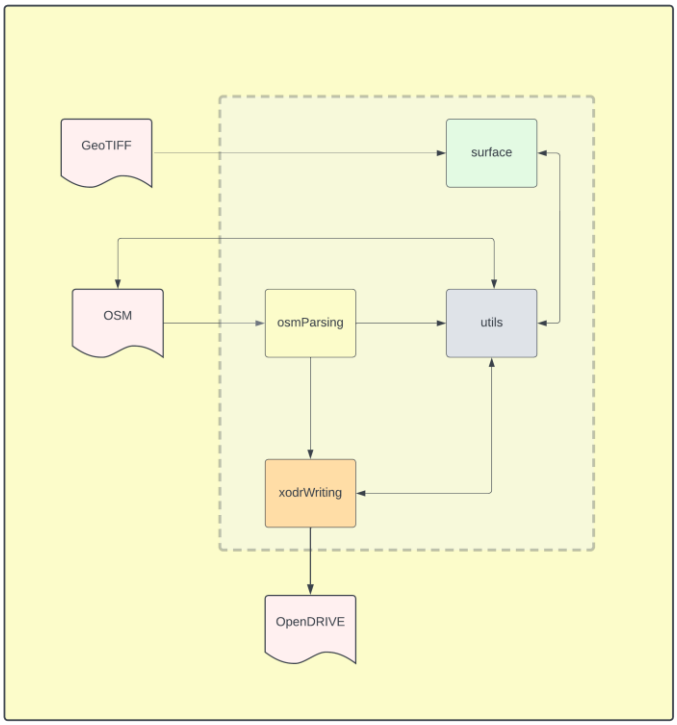

Figure 26: Diagram of the main tools

osmParsing Module

The osmParsing module provides the foundation for the converter's functionality. It is dedicated to parsing the complex OSM XML data and extracting relevant road network information. This module deciphers OSM elements including roads, lanes, intersections, and attributes.

surface Module

The surface module assumes the role of interfacing with GeoTIFF elevation data. This module reads and interprets the elevation information stored within GeoTIFF file. By systematically processing the GeoTIFF data, it constructs a surface a surface representation that accurately captures terrain height variations.

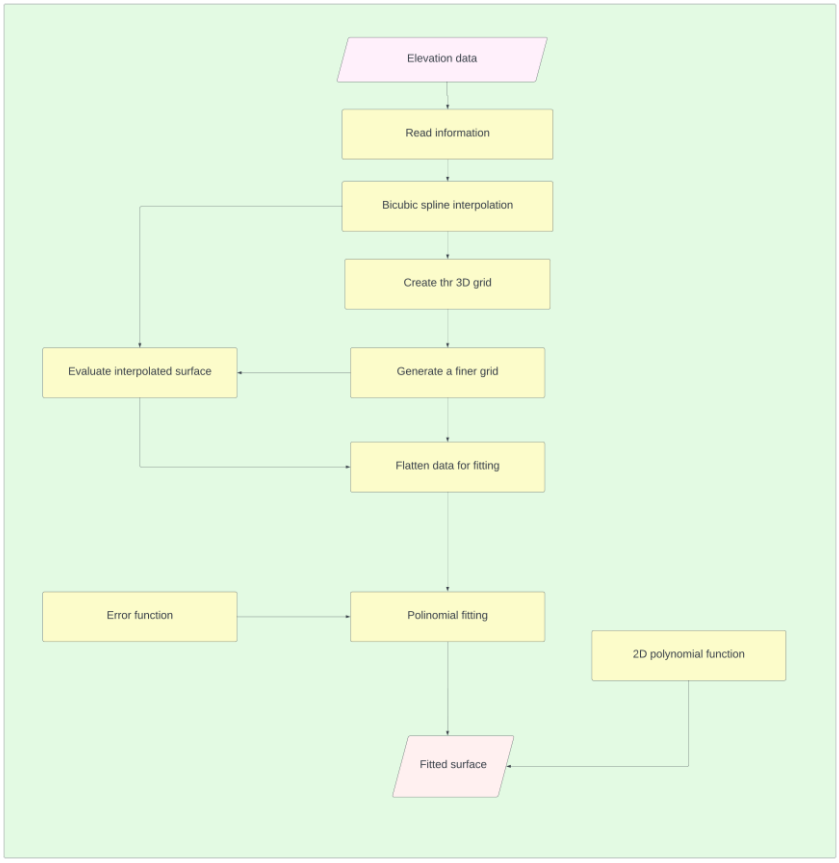

Figure 27: Diagram of the surface function

Creation of the grid

Creating a grid is necessary to establish a spatial coordinate system for the input data. The grid assigns coordinates to each data point, enabling interpolation, spatial analysis, and visualization. It provides a structured framework to organize, analyze, and represent spatial data accurately.

The process of creating the 2D grid involves obtaining the spatial bounds of the input data and generating a set of coordinates that span this spatial extent. This is accomplished using the src.bounds attribute and the numpy.linspace function.

The src.bounds attribute provides the minimum and maximum values of the x-coordinate (x_min, x_max) and y-coordinate (y_min, y_max) that define the spatial extent of the data.

A set of points must be defined along the x and y directions that span the spatial bounds. This is achieved by using the numpy.linspace function, which generates a sequence of evenly spaced values within a specified interval.

To the x-coordinate direction the starting point of the sequence('start') is defined as 'x_min', the ending point of the sequence('stop') is defined as 'x_max', and the number of points ('num') as the number of pixels or grid points in the input data. This generates a 1D array x with equally spaced points along the x-direction.

To the y-coordinate direction the starting point of the sequence('start') is defined as 'y_min', the ending point of the sequence('stop') is defined as 'y_max', and the number of points ('num') as the number of pixels or grid points in the input data. This generates a 1D array x with equally spaced points along the y-direction.

The two 1D arrays, x and y represent the coordinates of the grid points along the x and y directions respectively. Each element in these arrays corresponds to a specific pixel or data value in the input data.

To construct the 2D grid will be used the 'numpy.meshgrid' function. This function takes the 1D arrays x and y as input and returns two 2D matrices, 'xi' and 'yi'.

The 'xi' matrix is created by replicating the elements of the x array along the y-direction, resulting in a matrix where each row corresponds to the x values at a specific y-coordinate.

The ‘yi’ matrix is created by replicating the elements of the y array along the x-direction, resulting in a matrix where each column corresponds to the y values at a specific x-coordinate.

Together, they define a 2D coordinate system that spans the spatial extent of the input data, allowing for further computations and interpolation.

6.3.1. Surface generation

Interpolation

Interpolation methods are utilized to estimate values between known data points. In the case of study interpolation is needed to create a smooth and continuous surface representation based on the input data. The three main methods used for this kind of interpolation are: nearest neighbor interpolation, bilinear interpolation, and bicubic spline interpolation.

To accurately interpolate the data and retrieve the result of z based on x and y values, a 2D interpolation method was employed. This method enables simultaneous interpolation across both the x and y dimensions, ensuring a comprehensive assessment of the data.

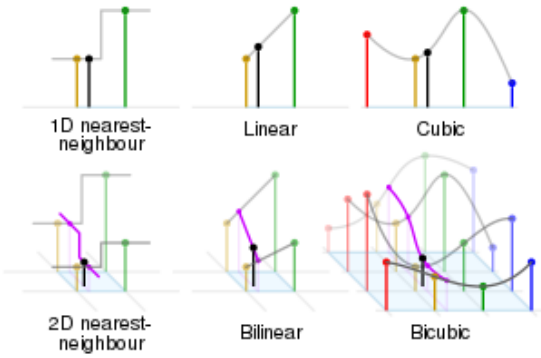

Figure 28: Comparison of some 1- and 2-dimensional interpolations

Nearest Neighbor Interpolation

Nearest neighbor interpolation is a simple and computationally efficient method used to estimate values between data points on a grid. It involves selecting the value of the nearest data point as the estimate for a target point. This technique is straightforward to implement and preserves the original data values, but it may result in a blocky or pixelated appearance due to its piecewise constant nature. Nearest neighbor interpolation is commonly used in applications where computational efficiency is critical, but it may not provide the highest level of accuracy or smoothness.

Bilinear Interpolation

Bilinear interpolation improves upon nearest neighbor interpolation by incorporating the surrounding data points to estimate values within a grid. It uses a weighted average of the neighboring four data points, considering the distance between the target point and these points. Bilinear interpolation produces smoother results than nearest neighbor interpolation and preserves linear trends in the data. This method is widely employed in image resizing, computer graphics, and spatial analysis applications. While bilinear interpolation provides better visual quality than nearest neighbor interpolation, it may not capture complex variations or sharp changes in the data.

Bicubic Spline Interpolation

Bicubic spline interpolation is a more sophisticated technique that achieves even higher accuracy and smoothness compared to bilinear interpolation. It constructs a smooth surface representation by fitting cubic polynomials to the surrounding data points within a grid. Bicubic spline interpolation considers 16 neighboring data points and employs a system of linear equations to determine the coefficients of the bicubic polynomial. This method provides a visually pleasing interpolation with continuous gradients and can capture intricate variations in the data. However, bicubic spline interpolation requires more computational resources than nearest neighbor or bilinear interpolation due to the increased complexity of the mathematical calculations involved.

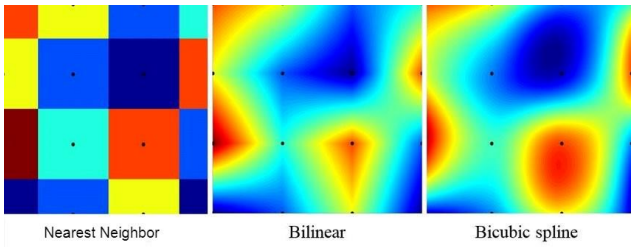

Figure 29 Results of 2D interpolation methods in a 2D respresentatrion

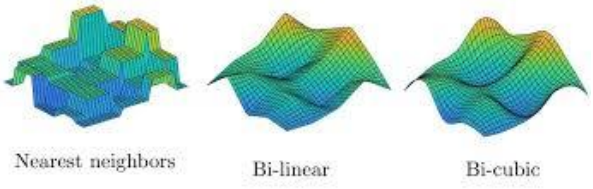

Figure 30 Results of the 2D interpolation methods in a 3D representation

	Nearest Neighbor Interpolation	**Bilinear Interpolation**	**Bicubic Spline Interpolation**
Interpolation Quality	Provides a piecewise constant approximation.	Produces a piecewise linear approximation.	Provides a smooth and continuous surface representation.
Accuracy	Low accuracy	Moderate accuracy	High accuracy
Smoothness	Can result in irregular surfaces, as it assigns the value of the nearest data point without considering neighboring points.	Can result in less smoothness, as it connects adjacent points with straight lines, leading to abrupt changes in slope	Provides a smooth surface approximation by employing cubic polynomials and ensuring continuity up to the second derivative
Handling Missing Data	Cannot handle missing data. Missing points are skipped, leading to gaps in the interpolated surface.	Cannot handle missing data. Missing points are skipped, leading to gaps in the interpolated surface.	Can handle missing data by incorporating neighboring points within the local region for interpolation.
Computational Complexity	Involves straightforward calculations, as it directly assigns the value of the nearest data point.	Involves simpler calculations, as it computes the weighted average of neighboring data points.	Involves more complex calculations, as it requires solving a system of equations to determine the polynomial coefficients.
Extrapolation	Cannot extrapolate beyond the original data boundaries. Interpolation is limited to the range of available data.	Cannot extrapolate beyond the original data boundaries. Interpolation is limited to the range of available data.	Can extrapolate beyond the original data boundaries, providing plausible estimates based on the fitted surface representation.
Applications	Commonly used in cases where preserving the original data values without interpolation is sufficient or when computational efficiency is the main concern.	Commonly used in simple data visualization tasks and basic spatial analysis where computational efficiency is prioritized over accuracy.	Widely used in computer graphics, image processing, GIS, scientific data analysis, and other fields where smooth and accurate surface representations are essential.

Table 7 Comparison of 2D interpolation methods

In summary, nearest neighbor interpolation is a simple and efficient method but may result in blocky artifacts. Bilinear interpolation offers smoother results by considering the surrounding data points, while bicubic spline interpolation provides the highest level of accuracy and smoothness by fitting cubic polynomials to a larger neighborhood of data points.

After comparing the different interpolation methods, it has been decided to utilize bicubic spline interpolation for creating the surface. Bicubic spline interpolation is a powerful technique that offers smoothness and accuracy. It extends cubic spline interpolation to two dimensions, providing a continuous and differentiable surface that closely fits the given data while minimizing oscillations. This choice ensures robustness and high-quality results in the interpolation process.

Bicubic interpolation shows up as a powerful and appropriate method for generating elevation surfaces due to its unique characteristics that align with the goals of the toolchain:

Smoothness and Continuity

Bicubic spline interpolation generates a smooth and continuous surface representation. It captures gradual changes in the data and avoids abrupt transitions between neighboring points. This property is crucial when creating a fitted surface to accurately represent the underlying data.

Interpolation Accuracy

Bicubic spline interpolation tends to provide higher accuracy compared to linear interpolation methods, especially when the underlying data has subtle variations or noise. By employing a more sophisticated mathematical approach, bicubic spline interpolation can better approximate the true values between data points.

Derivative Continuity

Bicubic spline interpolation ensures continuity of the interpolated surface's first and second derivatives. This property is advantageous in applications where derivatives of the surface, such as slope or curvature, are important for further analysis or visualization.

Compatibility with Grid Data

The bicubic spline interpolation method is well-suited for gridded data, which is the case in this function. It operates effectively on regular grids, allowing for interpolation at any point within the grid using neighboring data points. This property makes it suitable for working with raster data such as topographic maps.

Mathematical Foundation

Bicubic interpolation's mathematical foundation lies in the formulation of a continuous surface through a set of cubic polynomials. The interpolated surface f(x,y) can be expressed as:

$$f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

Equation 1: Equation of the interpolated surface

Where:

i, j: Indices that range from 0 to 3, representing the terms in the bicubic polynomial.

a_{ij}: Coefficients that need to be determined for accurate interpolation.

x^i: Powers of x corresponding to the current term i in the polynomial.

y^j: Powers of y corresponding to the current term j in the polynomial.

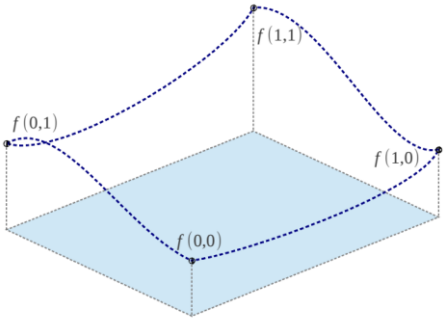

Figure 31: Representation of bicubic interpolation [56]

The primary challenge in bicubic interpolation is determining the 16 coefficients (a_{ij}) that will accurately describe the elevation surface between known data points. These coefficients encapsulate the information needed to create a smooth and continuous surface. The process of determining these coefficients is the heart of the interpolation technique.

This procedure yields a surface f(x,y) on the area of

$$f(x,y) = [x^3 \quad x^2 \quad x \quad 1] \begin{bmatrix} a_{3,3} & a_{3,2} & a_{3,1} & a_{3,0} \\ a_{2,3} & a_{2,2} & a_{2,1} & a_{2,0} \\ a_{1,3} & a_{1,2} & a_{1,1} & a_{1,0} \\ a_{0,3} & a_{0,2} & a_{0,1} & a_{0,0} \end{bmatrix} \begin{bmatrix} y^3 \\ y^2 \\ y \\ 1 \end{bmatrix}$$

Equation 2: Function for the determination of the interpolated surface

$$f(x,y) = \sum_{i=0}^{3} \sum_{j=0}^{3} a_{ij} x^i y^j$$

$$\partial_x f(x,y) = \sum_{i=1}^{3} \sum_{j=0}^{3} i a_{ij} x^{i-1} y^j$$

$$\partial_y f(x,y) = \sum_{i=0}^{3} \sum_{j=1}^{3} j a_{ij} x^i y^{j-1}$$

$$\partial_{xy} f(x,y) = \sum_{i=1}^{3} \sum_{j=1}^{3} ij a_{ij} x^{i-1} y^{j-1}$$

Equation 3: Partial derivative expressions for the computation of the surface

These derivatives provide insights into the surface's curvature and rate of change in both x and y directions. Equations that involve finite differences or explicit mathematical formulas are used to compute these derivatives.

For the computation of the bicubic spline interpolation as shown in Figure X the equations used will be finite difference approximations for calculating partial derivatives.

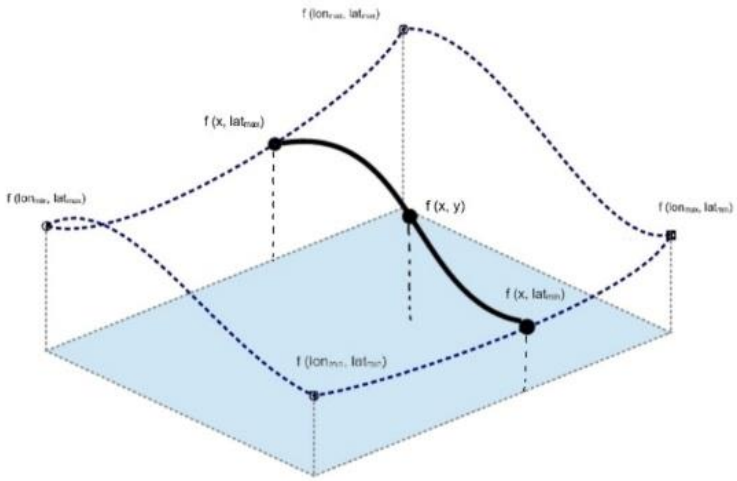

Figure 32:Representation of a bicubic spline interpolation

$$\partial_x f(x,y) = [f(x+1,y) - f(x-1,y)]/2$$
$$\partial_y f(x,y) = [f(x,y+1) - f(x,y-1)]/2$$
$$\partial_{xy} f(x,y) = [f(x+1,y+1) - f(x-1,y) - f(x,y-1) + f(x,y)]/4$$

Equation 4: Partial equations for the computation of the bicubic spline interpolation

By utilizing finite difference equations, we can obtain valuable insights into the behavior of the interpolated surface and use these approximations to refine the coefficients in the bicubic spline interpolation process, resulting in a more accurate representation of the elevation surface.

This step is performed in the code by utilizing the ‘RectBivariateSpline’ class from the ‘scipy.interpolate’ module.

6.3.2. Fitting method

Fitting a function to a surface involves determining a mathematical model or equation that approximates the relationship between the independent variables (denoted as x and y) and the dependent variable (denoted as z) within a two-dimensional space.

The primary goal of surface fitting is to capture and represent the underlying connections, trends, and patterns present in the data points. This process enables extrapolation beyond observed data to make predictions or to produce a smooth representation of the surface. Achieving this involves selecting an appropriate mathematical model that accurately reflects the data.

Several methods can be employed to find the optimal solution when fitting a function to a surface. These techniques aim to determine the model's parameters or coefficients that minimize the difference between the fitted surface and the observed data points. Commonly used methods include the least squares method, polynomial fitting, and spline interpolation. Each method offers unique advantages and is selected based on the specific characteristics of the data and the desired outcome.

Least Squares Optimization

Least squares optimization is a widely used method for fitting a function to a surface. It involves minimizing the sum of the squares of the differences between the fitted surface values and the corresponding observed data points. This approach provides a robust and balanced fit by considering all data points and giving more weight to larger discrepancies. The optimization process adjusts the parameters or coefficients of the function to iteratively reduce the objective function, resulting in the best-fitting solution. [57]

Polynomial Fitting

Polynomial fitting is a popular technique for approximating a surface with a polynomial function. The degree of the polynomial determines the complexity of the model and the number of coefficients to be determined. The coefficients are estimated by solving a system of linear equations using the least squares method or other numerical techniques. Polynomial fitting allows for flexible modeling of various surface shapes and can be extended to higher dimensions. [58]

Spline Interpolation

Spline interpolation is another powerful method for fitting a function to a surface. It involves constructing a smooth and continuous surface representation based on the observed data points. Various types of splines, such as cubic splines or B-splines, can be used to approximate the surface. The spline interpolation process determines the optimal coefficients or control points by solving a system of linear equations or applying optimization techniques to minimize the interpolation error.

Method	Advantages	Disadvantages	Computational Requirements
Least Squares Fitting	Provides an optimal fit according to the least squares criterion.	May struggle with complex surfaces exhibiting non-polynomial behavior.	Moderate computational requirements
	Effective in handling noise or errors in the data.	Requires defining an appropriate objective function.	
	Can be applied to various types of functions.		
Polynomial Fitting	Versatile technique that can capture various trends and patterns in the data.	Can result in overfitting if the degree of the polynomial is too high.	Low computational requirements
	Allows control over the complexity of the fit through the degree of the polynomial.	May not accurately capture complex surface behavior.	
Spline Interpolation	Effective in capturing complex variations in the data.	May require additional computational resources compared to simpler methods	Higher computational requirements
	Provides a visually pleasing fit with smooth transitions between adjacent data points.	Requires careful selection of spline parameters.	

Table 8 Comparation of the different fitting methods

The chosen approach for fitting the data is the least squares method. This method serves to determine the coefficients of a polynomial function that best align with the given dataset. Through the least squares optimization technique, the primary goal is to identify these coefficients in a manner that minimizes the disparity between the predicted values generated by the polynomial function and the actual data points.

The objective is to determine the polynomial coefficients that provide the best possible fit to the data. By minimizing the discrepancy between the predicted values and the observed data points, the least squares optimization aims to find the coefficients that yield the most accurate representation of the underlying relationship between the variables.

The least squares method is well-suited for this purpose. It aims to minimize the sum of the squared differences between the predicted values of the polynomial

function and the actual data points on the finer grid. By minimizing the squared residuals, the method provides an optimal fit that reduces the overall error between the fitted surface and the observed data.

Implementation

The ‘least_squares’ function is called to perform the least squares optimization. It takes several arguments: the error function (error_func), the initial coefficient values (initial_coeffs), and the flattened data (x_flat, y_flat, z_flat).

The ‘error_func’ is defined as a function that quantifies the difference between the predicted values of the polynomial function and the observed data points. This error function computes the residual, which represents the discrepancy between the polynomial fit and the actual data.

Concurrently, an initial guess for the polynomial coefficients is provided using the ‘initial_coeffs’ variable. These initial values serve as the starting point for the optimization process.

The ‘least_squares’ optimization algorithm iteratively adjusts the polynomial coefficients to minimize the error function. By minimizing the sum of squared differences between the predicted polynomial values and the actual data points, it finds the optimal values for the coefficients that provide the best fit to the data.

Once the optimization process is completed, the result contains the optimal values for the polynomial coefficients. These coefficients represent the parameters of the fitted polynomial function that provides the best approximation to the data.

In conclusion, this approach enables the derivation of a polynomial function that accurately represents the underlying relationship between the variables, improving the understanding and modeling of the data.

$$m_x \in \begin{cases} x_L \\ x_{L+1} \end{cases} \begin{cases} h_{xL} \\ h_{xL+1} \end{cases} \frac{h_{xL} - h_{xL+1}}{x_L - x_{L+1}}$$

Equation 5: Calculation of the slope between two adjacent points on x coordinates

$$m_y \in \begin{cases} y_L \\ y_{L+1} \end{cases} \begin{cases} hy_L \\ h_{yL+1} \end{cases} \frac{h_{yL} - h_{yL+1}}{y - y_{L+1}}$$

Equation 6:Calculation of the slope between two adjacent points on y coordinates

$$\Delta x = x_L - int(x_L)$$

Equation 7: Differential of x

$$\Delta y = y_L - int(y_L)$$

Equation 8: Differential of y

$$heigth = \begin{bmatrix} h_x \\ h_y \end{bmatrix} = \begin{bmatrix} m_x * \Delta x + h_{xL} \\ m_y * \Delta y + h_{yL} \end{bmatrix}$$

Equation 9: Linear interpolation between two points

Methodology

The developed algorithm employs bilinear interpolation for computing the height value. This interpolation method is chosen for its effectiveness and accuracy in approximating the height between four neighboring points in a grid.

Algorithm 1 Computation bilinear interpolation

```
res_x ← map width / (map width - min.  x coord.)
res_y ← map heigth / (max.  y cood.  - min.  y coord.)
m_x ← (h - h_n) / res_x
m_y ← (h - h_n) / res_y
x_diff ← x coord.  - norm.  x
y_diff ← y coord.  - norm.  y
b_diff ← h
height ← ((m_y * y_diff) + (m_x * x_diff) + b_diff)
```

Figure 33: Pseudocode of the algorithm for the computation of the bilinear interpolation

6.3.3. Comparison of the order of the function

Comparing the orders of functions used for surface fitting is crucial for understanding both the complexity and accuracy of the model employed to represent the surface. The selection of an appropriate order aims to strike a delicate balance, seeking the most precise mathematical depiction while avoiding overfitting and excessive computational demands.

Higher-order functions hold promise for capturing intricate data patterns with greater fidelity. However, they also introduce the risk of overfitting, where the model becomes overly attuned to the training data, hindering its ability to generalize to new observations. Conversely, lower-order functions offer simplicity but may lack the capacity to capture the full complexity of the data, resulting in underfitting.

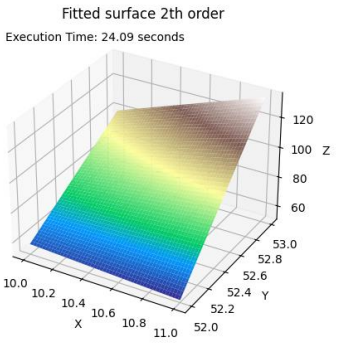

Figure 34: Representation of the surface with an 2nd order fitment

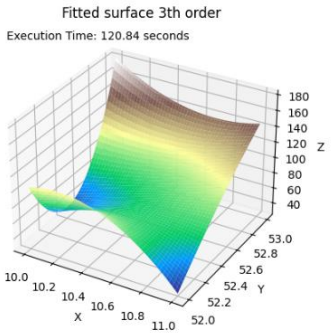

Figure 35: Representation of the surface with an 3th order fitment

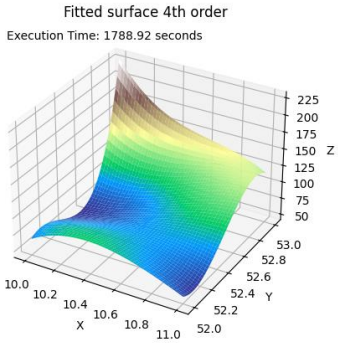

Figure 36: Representation of the surface with an 4th order fitment

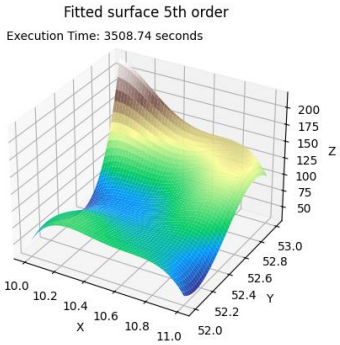

Figure 37: Representation of the surface with an 5th order fitment

Upon comparing the orders of functions depicted in the provided figures, a discernible difference becomes apparent. In the case at hand, a fourth-order function emerged as the preferred solution. It strikes an optimal balance, offering sufficient precision to capture the intricate geography while mitigating the risk of overfitting and conserving computational resources.

Algorithm 2 Generation of the elevation surface

Require: tif_file

function FITTED_SURFACE
  $x \leftarrow b\_left \quad b\_right \quad ele\_map\_width$
  $y \leftarrow b\_bottom \quad b\_top \quad ele\_map\_height$
  $fine\_x \leftarrow x \; b\_left \quad b\_right \quad ele\_map\_width \times 3$
  $fine\_y \leftarrow y \; b\_bottom \quad b\_top \quad ele\_map\_height \times 3$
  $spline \leftarrow \text{RectBivariateSpline}(y, x, tif\_data)$

  $fine\_xi, fine\_yi \leftarrow \text{meshgrid}(fine\_x, fine\_y)$

  $zi \leftarrow spline|_{fine\_yi, fine\_xi}$

  $x\_flat \leftarrow$ fine_xi.flatten()
  $y\_flat \leftarrow$ fine_yi.flatten()
  $z\_flat \leftarrow$ zi.flatten()

  function POLY_FUNC($coeffs, x, y$)
    $n \leftarrow \text{int}(\sqrt{n\_coeff})$
    return $\sum$ coeffs_index $\times\, x^i \times y^j$ for $i \in \text{range}(n)$, $j \in \text{range}(n)$
  end function

  function ERROR_FUNC($coeffs, x, y, z$)
    return POLY_FUNC($coeffs, x, y$) - z
  end function

  $$initial\_coeffs \leftarrow \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

  $result \leftarrow \text{least\_squares}(\text{error\_func}, \text{initial\_coeffs}, \text{args})$

  $fitted\_surface \leftarrow \text{poly\_func}(\text{result.x}, \text{fine\_xi}, \text{fine\_yi})$

  return $fitted\_surface$

end function

Figure 38: Pseudocode of the algorithm for the generation of the surface

The algorithm outlined serves as a fundamental component in the generation of elevation surfaces. With the requirement of a TIFF file containing elevation data, the algorithm begins by initializing variables to define the boundaries of the elevation map. These variables encompass the horizontal and vertical extents, delineating the region over which the surface will be generated.

Utilizing spline interpolation, specifically the RectBivariateSpline method, the algorithm constructs a smooth surface representation of the elevation data. This interpolation technique ensures that the generated surface accurately captures the underlying terrain features while mitigating the effects of noise or irregularities present in the data.

Following the spline interpolation, the algorithm proceeds to generate a fine meshgrid, refining the resolution of the surface representation. This finer grid facilitates more precise evaluation of the elevation values, essential for achieving high-fidelity surface reconstructions.

Subsequently, the algorithm evaluates the spline interpolation on the fine meshgrid, yielding elevation values across the specified region. These values serve as the basis for fitting a polynomial function, aimed at capturing the underlying trends and variations in the elevation data.

The process of polynomial fitting involves minimizing the error between the calculated polynomial and the actual elevation values. Leveraging least squares optimization, the algorithm iteratively adjusts the polynomial coefficients to achieve the best possible fit to the data. This optimization technique ensures that the fitted surface closely aligns with the observed elevation data, enhancing the accuracy and reliability of the generated surface.

Upon obtaining the optimal polynomial coefficients, the algorithm calculates the fitted surface, synthesizing the polynomial function with the fine meshgrid coordinates. This step yields a comprehensive representation of the terrain, characterized by smooth transitions and accurately captured features.

Finally, the algorithm concludes by returning the fitted surface as the output, providing a valuable resource for subsequent analyses and applications. By integrating advanced interpolation and polynomial fitting techniques, the algorithm facilitates the generation of elevation surfaces that are both precise and reliable.

6.3.4. Generation of the OpenDrive file

As mentioned in chapter 5, the OpenDrive format follows a hierarchical structure with each level serving a specific pourpuse in describing the road environment for autonomous vehicles.

At the top of the hierarchy, we have as a foundation for the file the road level. This provides the overall characteristics of the roads. These road definitions act as the starting point, forming the basis for the entire representation. For a better understanding of the concept the structure of the roads is represented in Figure 39.

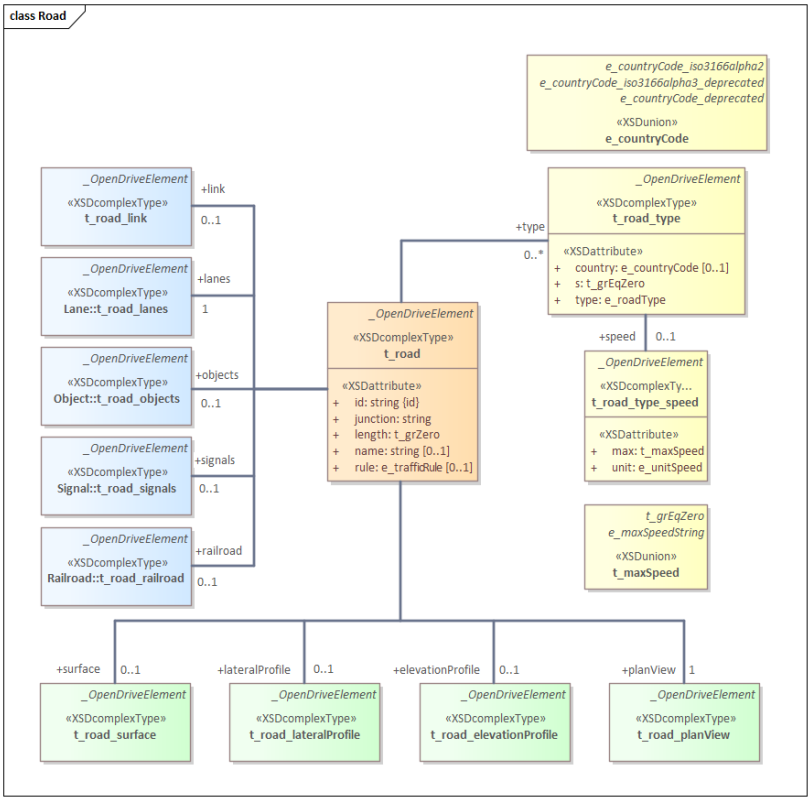

Figure 39: UML of the road class in OpenDrive maps

As we move down the hierarchy, we encounter the lanes. Lanes are like individual lanes on a real-world road. They encompass crucial details like lane width, road markings, and even whether a lane supports certain driving behaviors. Think of lanes as the distinct paths that vehicles can travel on within the virtual road network.

In our virtual road system, there are also places where lanes intersect and interact – much like intersections. Just as real-world junctions enable vehicles to change paths, our virtual junctions define how lanes connect and how vehicles can navigate from one lane to another. These junctions ensure a seamless flow of traffic in the digital environment.

But the aim of this thesis is to focus on the generation of the elevation profile. For this part of the file, we will focus on the elevationProfile element inside the road class.

The elevation profile in OpenDrive format is defined by a cubic equation with the following mathematical formula:

$$elev(ds) = a + b * ds + c * ds^2 + d * ds^3$$

Equation 10: Polynomial function used to model elevation variation

where:

elev	is the elevation at a given position.
a,b,c,d	are polynomial coefficients.
ds	is the distance along the reference line between the start of a new elevation element and the given position.

The ds value restarts at 0 for each element. The absolute position of an elevation value is computed using the following equation:

$$s = s_{start} + ds$$

Equation 11: Calculation of a new position s based on the initial position

where:

s	is the absolute position in the reference line coordinate system.
Sstart	is the start position of the element on the reference line coordinate system.

Algorithm 3 Computation of the normalized coordinates for bilinear interpolation

```
if topomap is not None then
    if not minRemoved then
        f_x_lookup ← (map_width × (x − min_x)/(max_x − min_x)
        x_lookup ← int(f_x_lookup)
        f_y_lookup ← (mapheight × (1.0 − (y − min_y)/(max_y − min_y)
        y_lookup ← int(f_y_lookup)
    else
        f_x_lookup ← (mapwidth × x/(maxx − coordinate − minx − coordinate)
        x_lookup ← int(f_x_lookup)
        f_y_lookup ← (mapheight × (1.0 − (y/(maxy − coordinate − miny − coordinate)
        y_lookup ← int(f_y_lookup)
    end if
end if
```

Figure 40:Pseudocode of the algorithm for the computation of the normalized coordinates for bilinear interpolation

The provided algorithm is a key component in the complex process of adding elevation data to an HD map.

The algorithm starts with a precondition check, which ensures the presence of a valid topographic map ('topomap'). This thorough procedure checks for errors caused by data incompleteness or corruption. By certifying the map's integrity, the technique lays the groundwork for future recovery of reliable elevation data.

Followed the variable `minRemoved` acts as a dynamic switch, indicating whether certain minimum values have been omitted from the dataset. Leveraging this adaptive mechanism, the algorithm showcases its versatility, adeptly maneuvering through varied datasets and scenarios with finesse and accuracy.

The algorithm's core goal is to compute normalized coordinates, which are required for further interpolation procedures. When'minRemoved' returns false, the algorithm uses special formulas to generate 'f x lookup' and 'f y lookup'. These calculations intelligently scale the input coordinates ('x' and 'y') to match the map's dimensions. This normalization procedure is critical, since it ensures consistency in the depiction of elevation data over the whole map canvas.

Conversely, when the presence of deleted minimum values is recognized ('minRemoved' is true), the method adjusts its calculations. It dynamically adjusts the normalization algorithms to account for the changed dataset features, assuring elevation representation integrity in the absence of specific data points. This adaptability demonstrates the algorithm's ability to smoothly accommodate varied datasets and developing settings.

Post normalization, the algorithm seamlessly transitions to the conversion phase, where it elegantly transforms the floating-point normalized coordinates into integer indices (`x_lookup` and `y_lookup`). These indices serve as navigational beacons, efficiently guiding the subsequent retrieval of elevation data from the HD map. The precision of this conversion process is pivotal, ensuring seamless compatibility with array indexing and facilitating expedited data access and manipulation operations.

In summation, the algorithm assumes a pivotal role in the intricate orchestration of furnishing elevation data to an HD map. Its adaptive nature, coupled with the precision in computing normalized coordinates and seamless data retrieval mechanisms, culminate in the creation of meticulous and accurate terrain

representations. Seamlessly integrated into the HD map ecosystem, the algorithm underpins a plethora of geographic and navigational applications, fortifying their reliability and efficacy in real-world scenarios.

6.4. Comparative analysis of virtual and real-world elevation data

Elevation Map Preparation

The real-world elevation map utilized in this analysis comes from the webpage FloodMap.net. This website offers a range of geographical data, including elevation maps that vividly depict varying elevations through a color-coded scheme. For the purpose of this study, the elevation map specifically pertains to the geographic region of Wolfenbüttel.

Figure 41: Colored topogrphic map of Wolfenbüttel from topographic-map.com [59]

Virtual Model Output Inspection

To retrieve the elevation data from the virtual model, we relied on the libOpenDrive library, accessible via a GitHub repository. This comprehensive library offers a range of functionalities for parsing OpenDrive files and extracting diverse geographical and topographical data, including elevation details. The primary aim was to leverage the OpenDrive viewer module within the library, enabling to engage with and visualize the elevation values of the virtual model more effectively.

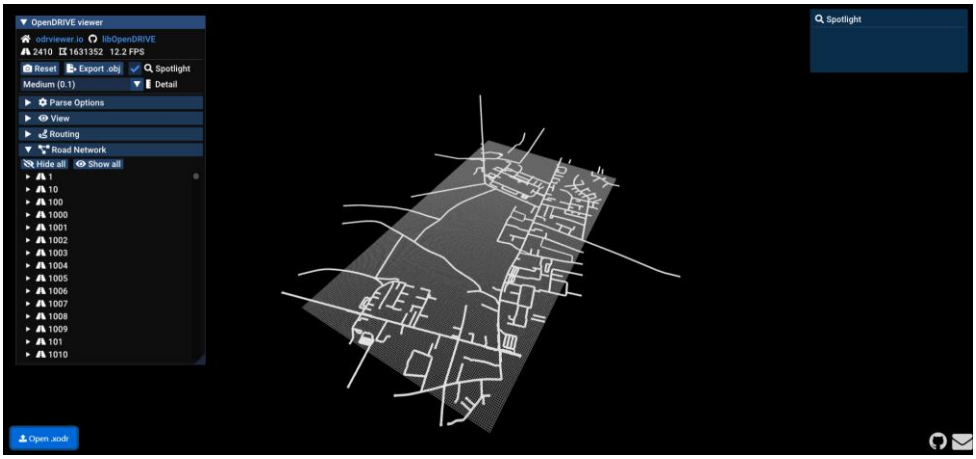

Figure 42: Representation of the OpenDrive map on OpenDRIVE viewer [60]

The OpenDrive viewer offered not only visualize the elevation data itself but also to simultaneously represent the corresponding x, y, and z coordinates. This dynamic visualization enhanced our understanding of the elevation distribution and its relationship to geographic positioning within our chosen area of study.

Using the tool's interactive interface, navigation through the virtual model's representation was facilitated, allowing for the identification of particular geographic points of interest, including Am Exer, Neuer Weg, and Jahnstraße. Utilizing the OpenDrive viewer, elevation data corresponding to these points was accurately extracted, providing a detailed overview of the virtual model's elevation outputs.

This extraction method not only visualized elevation values but also contextualized them with geographical coordinates. By accurately capturing

elevation data from selected points, we established a solid foundation for subsequent visual comparisons and manual assessments of the virtual model's representation.

Visual Comparison

The visual comparison entails placing the elevation representation of the virtual model and the real-world elevation map side by side for direct scrutiny. This allows for an immediate and comprehensive assessment of how the virtual model's elevation outputs correspond to the elevation representation in reality. By juxtaposing these representations, we can identify areas of agreement and divergence more effectively.

During the visual comparison, our attention is directed towards observing regions where the virtual model's elevation representation aligns closely with the real-world elevation map, as well as regions where noticeable differences become apparent. These areas of alignment and discrepancies serve as essential markers for evaluating the accuracy and fidelity of the virtual model's elevation outputs.

To gain a deeper understanding of the comparison, we pay particular attention to distinctive features within the elevation representations. The main focus of the case study is to ensure that the Salzdahlumstaße road has a constant gradient from Neuer Weg to Am Exer. These distinctive feature can provide valuable insights into how well the virtual model captures the nuanced topographical characteristics of the real-world terrain.

Manual Annotation

The manual annotation process serves as a meticulous means to highlight and document variations in elevation representation. By incorporating notations, comments, or markings, we aim to precisely delineate the regions of interest where differences are evident. This manual annotation provides a tangible record of our observations, aiding in the subsequent analysis and interpretation of the comparison results.

During the visual comparison, we pay particular attention to areas where differences emerge between the virtual model's elevation representation and the real-world elevation map. These differences can manifest as discrepancies in elevations, terrain contours, or topographical characteristics. We manually annotate these areas, marking them with relevant notations, comments, or visual cues to pinpoint the locations and extent of differences.

In addition to indicating the presence of differences, we strive to assess the magnitude and nature of these variations. This entails considering whether differences are consistent across multiple points, indicative of systematic biases, or if they occur sporadically, indicating unexpected variations. By comprehensively analyzing the nature of differences, we gain insights into potential factors influencing the alignment between the virtual model and reality.

The manual annotations and markings created during this process serve as a tangible record of our visual observations. These annotations become invaluable assets for subsequent analysis, enabling us to identify trends, patterns, and potential correlations in the discrepancies between the virtual model's elevation representation and the real-world elevation map.

As we conduct the visual comparison, we meticulously note and document our observations. Both areas of agreement and differences are recorded, along with any notable patterns or trends identified during the comparison. These notations form the foundation for subsequent analysis and conclusions.

Results and Analysis

Record the insights gained from the visual comparison process. Document observations, patterns, and discrepancies that stand out. Note any areas where the virtual model's elevation representation closely matches the real-world elevation map and areas where differences are more pronounced.

Error Patterns

Identify any consistent error patterns that arise from the comparison. These could include systematic overestimation or underestimation of elevations, misalignment of features, or unexpected variations in elevation.

RoadId	z_virtual	z real
404	92	85
37	96	96
51	99	100
266	99	127

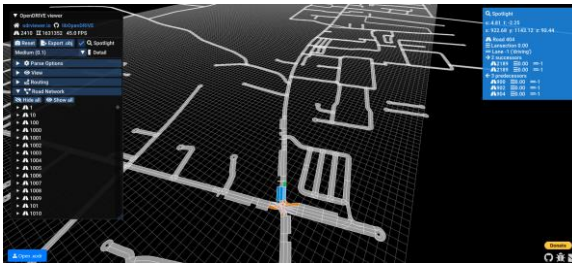

Figure 43: Digital representation of the crossway between Ahlumer Straße and Neuer Weg

Figure 44: Topographic map of Wolfenbüttel of the crossway between Ahlumer Straße and Neuer Weg

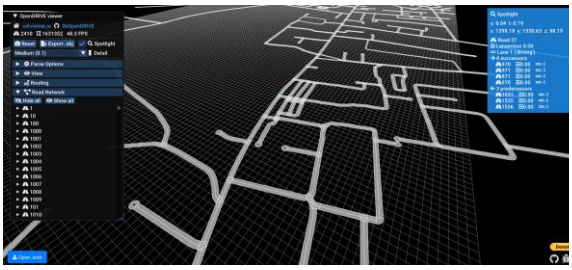

Figure 45: Digital representation of Ostfalia Hochschule Parking 1

Figure 46: Topographic map of Ostfalia Hochschule Parking 1

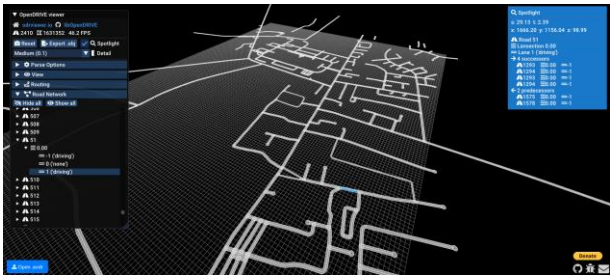

Figure 47: Digital representation of Ostfalia Hochschule Parking 2

Figure 48: Topographic map of Ostfalia Hochschule Parking 2

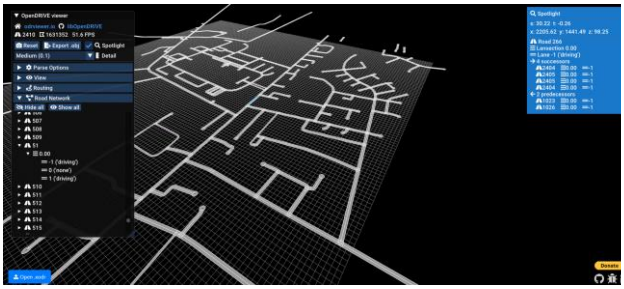

Figure 49: Digital representation of Am Exer Buildings

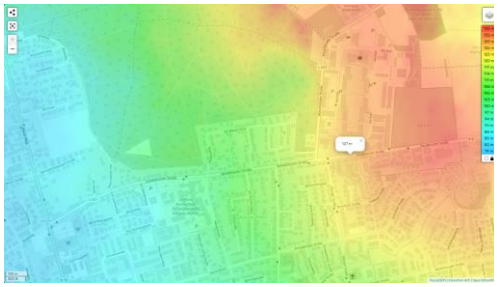

Figure 50: Topographic map of Am Exer Buildings

Implications and Validation

Assessment of Accuracy

When comparing our virtual model to real-world elevation data, we noticed some differences, especially in the area of Am Exer. The virtual model consistently showed lower elevations than what actually exists.

This means that while our model is mostly accurate, there are areas where it needs improvement, especially in accurately representing elevations. To make our model more reliable, we need to figure out why these discrepancies occur and find ways to fix them. This might involve using better algorithms or getting more accurate data.

In summary, while our virtual model is good in most places, we need to keep working on it to ensure it's accurate everywhere, especially in places like Am Exer.

The manual visual comparative analysis presented in this chapter offers a unique perspective on evaluating elevation data from a virtual model. By directly engaging with the data. While manual comparison requires careful observation and interpretation, it provides valuable insights that complement automated methods and enhance the assessment of virtual modeling techniques.

7. Regulations and Standards in HD Map Generation

In the pursuit of safe and reliable autonomous driving systems, the creation and utilization of High Definition (HD) maps have emerged as a critical component. The generation of accurate and up-to-date HD maps requires adherence to a range of regulations and standards to ensure functional safety, interoperability, and consistency. This chapter delves into the regulatory and standardization landscape surrounding HD map generation for autonomous driving, examining key norms that guide the industry toward the realization of safe and efficient self-driving vehicles.

7.1. Regulations

7.1.1. ISO 26262-1:2018

ISO 26262-1:2018, [61] an internationally recognized standard, focuses on functional safety for road vehicles. It delineates processes and requirements to manage functional safety risks inherent in electrical and electronic systems within vehicles, including those involved in autonomous driving. This section explores the applicability of ISO 26262 to HD map generation systems, highlighting how safety concerns are addressed to ensure robust and dependable HD map data.

7.1.2. SAE J3016

SAE J3016 [62] classifies driving automation into six levels, from Level 0 (no automation) to Level 5 (full automation). HD maps play a pivotal role in the higher levels of automation, where vehicles must possess a comprehensive understanding of their environment. This portion of the chapter explores the importance of HD map data in enabling advanced levels of autonomy and how

SAE J3016 categorization guides the integration of HD maps in autonomous driving systems.

7.1.3. Local Regulations and Industry Guidelines

Autonomous driving and HD map generation are subject to evolving regional regulations and industry guidelines. This section investigates the role of regulatory bodies like the U.S. National Highway Traffic Safety Administration (NHTSA) and the European Union in shaping norms for autonomous vehicles. Additionally, the application of industry-specific norms such as the Automotive Safety Integrity Level (ASIL) process is examined to ensure safety and adherence to rigorous standards.

7.2. Standards

7.2.1. ASAM OpenDRIVE

ASAM OpenDRIVE stands as a significant standard for road and lane data representation. It provides an open and standardized XML format for describing road networks and environments, a fundamental necessity for HD map generation. This section delves into how ASAM OpenDRIVE facilitates the consistent representation of road geometries and environments, enabling autonomous vehicles to comprehend their surroundings accurately.

8. New lines of research

As the field of autonomous driving rapidly evolves, the demand for cutting-edge tools and methodologies to simulate and evaluate autonomous vehicle behavior within complex environments intensifies.

Integration of Real-Time Sensor Data

One interesting path is the incorporation of real-time sensor data into the created digital environments. Research in this direction entails developing techniques to seamlessly fuse virtual and real data, allowing for more accurate validation of autonomous systems. Exploring the challenges and opportunities of integrating dynamic sensor data within the toolchain can contribute to more robust and realistic simulations.

Generation of digital environments

Current tools struggle to accurately capture the complexities of junctions, leading to errors in navigation systems and autonomous vehicle simulations. This new research could explore better algorithms for modeling junction geometry and topology, incorporate realistic traffic behavior simulation, and leverage machine learning for junction detection and classification. Additionally, involving human feedback and crowdsourced data could help improve junction representations.

9. Conclusions

In conclusion, the development of a toolchain for the automated generation of digital maps for autonomous drive represents a significant advancement in the field of autonomous vehicle technology.

Through the integration of the elevation data, this toolchain not only offers a streamlined and efficient solution for creating high-definition maps, essential for safe and reliable autonomous navigation, but also fullfills the gap in the research world due to there wasn't any study related with the annotation of the elevation on OpenDRIVE maps.

As autonomous driving continues to evolve, this toolchain serves as a foundational framework, enabling further advancements in autonomous vehicle technology and paving the way for a future of safer and more efficient transportation systems.

10. References

[1] U. Spiegelhalter, “Annotating OpenStreetMap data with elevation data”.

[2] “ASAM OpenDRIVE.” Accessed: Oct. 22, 2023. [Online]. Available: https://www.asam.net/index.php?eID=dumpFile&t=f&f=4422&token=e590561f3c39aa2260e5442e29e93f6693d1cccd

[3] NASA JPL, “NASADEM Merged DEM Global 1 arc second V001.” NASA EOSDIS Land Processes Distributed Active Archive Center, 2020. doi: 10.5067/MEASURES/NASADEM/NASADEM_HGT.001.

[4] J.-H. Meusener, “JHMeusener/osm2xodr.” Feb. 29, 2024. Accessed: Mar. 03, 2024. [Online]. Available: https://github.com/JHMeusener/osm2xodr

[5] C.-Y. Chan, “Advancements, prospects, and impacts of automated driving systems,” *Int. J. Transp. Sci. Technol.*, vol. 6, no. 3, pp. 208–216, Sep. 2017, doi: 10.1016/j.ijtst.2017.07.008.

[6] K. Jo and M. Sunwoo, “Generation of a Precise Roadway Map for Autonomous Cars,” *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 3, pp. 925–937, Jun. 2014, doi: 10.1109/TITS.2013.2291395.

[7] R. Vivacqua, R. Vassallo, and F. Martins, “A Low Cost Sensors Approach for Accurate Vehicle Localization and Autonomous Driving Application,” *Sensors*, vol. 17, no. 10, p. 2359, Oct. 2017, doi: 10.3390/s17102359.

[8] R. Liu, J. Wang, and B. Zhang, “High Definition Map for Automated Driving: Overview and Analysis,” *J. Navig.*, vol. 73, no. 2, pp. 324–341, Mar. 2020, doi: 10.1017/S0373463319000638.

[9] D. Pannen, M. Liebner, W. Hempel, and W. Burgard, “How to Keep HD Maps for Automated Driving Up To Date,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, Paris, France: IEEE, May 2020, pp. 2288–2294. doi: 10.1109/ICRA40945.2020.9197419.

[10] “How medium-definition maps help navigate dynamic roads - GPS World,” GPS World - The Business and Technology of Global Navigation and Positioning. Accessed: Oct. 29, 2023. [Online]. Available: https://www.gpsworld.com/how-medium-definition-maps-help-navigate-dynamic-roads/

[11] X. Wang *et al.*, “HD Map Construction and Update System for Autonomous Driving in Open-Pit Mines,” *IEEE Syst. J.*, pp. 1–12, 2023, doi: 10.1109/JSYST.2023.3317288.

[12] H. Wang, C. Xue, Y. Zhou, F. Wen, and H. Zhang, “Visual Semantic Localization based on HD Map for Autonomous Vehicles in Urban Scenarios,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11255–11261. doi: 10.1109/ICRA48506.2021.9561459.

[13] T. Qin, Y. Zheng, T. Chen, Y. Chen, and Q. Su, “A Light-Weight Semantic Map for Visual Localization towards Autonomous Driving,” in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 11248–11254. doi: 10.1109/ICRA48506.2021.9561663.

[14] M. Bäumker and F. Heimes, “New Calibration and Computing Method for Direct Georeferencing of Image and Scanner Data Using the Position and Angular Data of an Hybrid Inertial Navigation System,” *Proc. OEEPE Workshop Integr. Sens. Orientat.*, Jan. 2002.

[15] M. Barsi and A. Barsi, “BUILDING OPENDRIVE MODEL FROM MOBILE MAPPING DATA,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLIII-B4-2021, pp. 9–14, Jun. 2021, doi: 10.5194/isprs-archives-XLIII-B4-2021-9-2021.

[16] “What Is Object Detection?” Accessed: Mar. 03, 2024. [Online]. Available: https://de.mathworks.com/discovery/object-detection.html

[17] G. Csurka, “An Efficient Approach to Semantic Segmentation,” *Int. J. Comput. Vis.*, vol. 95, pp. 198–212, Nov. 2011, doi: 10.1007/s11263-010-0344-8.

[18] S. A. Bello *et al.*, “Cloud computing in construction industry: Use cases, benefits and challenges,” *Autom. Constr.*, vol. 122, p. 103441, Feb. 2021, doi: 10.1016/j.autcon.2020.103441.

[19] “4D LiDARs vs 4D RADARs — Why the LiDAR vs RADAR comparison is more relevant today than ever,” Welcome to The Library! Accessed: Mar. 03, 2024. [Online]. Available: https://www.thinkautonomous.ai/blog/fmcw-lidars-vs-imaging-radars/

[20] R. Zhang and K. Cai, “The Application of Edge Computing in High-Definition Maps Distribution,” Sep. 2020, pp. 116–121. doi: 10.1145/3425329.3425333.

[21] V. Ilci and C. Toth, “High Definition 3D Map Creation Using GNSS/IMU/LiDAR Sensor Integration to Support Autonomous Vehicle Navigation,” *Sensors*, vol. 20, no. 3, 2020, doi: 10.3390/s20030899.

[22] I. Bilik, O. Longman, S. Villeval, and J. Tabrikian, “The Rise of Radar for Autonomous Vehicles: Signal Processing Solutions and Future Research Directions,” *IEEE Signal Process. Mag.*, vol. 36, no. 5, pp. 20–31, Sep. 2019, doi: 10.1109/MSP.2019.2926573.

[23] M. Elhashash, H. Albanwan, and R. Qin, “A Review of Mobile Mapping Systems: From Sensors to Applications,” *Sensors*, vol. 22, no. 11, Art. no. 11, Jan. 2022, doi: 10.3390/s22114262.

[24] D. Tosic, S. Tuttas, L. Hoegner, and U. Stilla, “FUSION OF FEATURE BASED AND DEEP LEARNING METHODS FOR CLASSIFICATION OF MMS POINT CLOUDS,” *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-2/W16, pp. 235–242, Sep. 2019, doi: 10.5194/isprs-archives-XLII-2-W16-235-2019.

[25] F. Poggenhans, N. O. Salscheider, and C. Stiller, “Precise Localization in High-Definition Road Maps for Urban Regions,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid: IEEE, Oct. 2018, pp. 2167–2174. doi: 10.1109/IROS.2018.8594414.

[26] J. Rebut, A. Ouaknine, W. Malik, and P. Perez, “Raw High-Definition Radar for Multi-Task Learning,” in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, New Orleans, LA, USA: IEEE, Jun. 2022, pp. 17000–17009. doi: 10.1109/CVPR52688.2022.01651.

[27] Y. Zhang, A. Carballo, H. Yang, and K. Takeda, “Perception and sensing for autonomous vehicles under adverse weather conditions: A survey,” *ISPRS J. Photogramm. Remote Sens.*, vol. 196, pp. 146–177, Feb. 2023, doi: 10.1016/j.isprsjprs.2022.12.021.

[28] D. J. Yeong, G. Velasco-Hernandez, J. Barry, and J. Walsh, “Sensor and Sensor Fusion Technology in Autonomous Vehicles: A Review,” *Sensors*, vol. 21, no. 6, Art. no. 6, Jan. 2021, doi: 10.3390/s21062140.

[29] J. Kim, D. S. Han, and B. Senouci, “Radar and Vision Sensor Fusion for Object Detection in Autonomous Vehicle Surroundings,” in *2018 Tenth International Conference on Ubiquitous and Future Networks (ICUFN)*, Prague: IEEE, Jul. 2018, pp. 76–78. doi: 10.1109/ICUFN.2018.8436959.

[30] “A fog blog: Understanding a challenge inherent to driving in San Francisco,” Waymo. Accessed: Mar. 03, 2024. [Online]. Available: https://waymo.com/blog/2021/11/a-fog-blog

[31] “9 Types of Sensor Fusion Algorithms,” Welcome to The Library! Accessed: Mar. 03, 2024. [Online]. Available: https://www.thinkautonomous.ai/blog/9-types-of-sensor-fusion-algorithms/

[32] A. Basiri, P. Amirian, and P. Mooney, “Using Crowdsourced Trajectories for Automated OSM Data Entry Approach,” *Sensors*, vol. 16, no. 9, Art. no. 9, Sep. 2016, doi: 10.3390/s16091510.

[33] F.-J. Behr, P. A.P., M. Ngigi, M. Zimmermann, and M.(Editors), *Geoinformation for a better World*. 2011.

[34] P. Mooney and M. Minghini, “A review of OpenStreetMap data,” *Mapp. Citiz. Sens.*, pp. 37–59, 2017.

[35] “OSM data model | OpenGeoEdu.” Accessed: Oct. 22, 2023. [Online]. Available: https://learn.opengeoedu.de/en/opendata/vorlesung/freiwillig-erhobene-daten/openstreetmap/datenmodell

[36] “Node - OpenStreetMap Wiki.” Accessed: Oct. 22, 2023. [Online]. Available: https://wiki.openstreetmap.org/wiki/Node

[37] “Altitude - OpenStreetMap Wiki.” Accessed: Nov. 07, 2023. [Online]. Available: https://wiki.openstreetmap.org/wiki/Altitude

[38] “Way - OpenStreetMap Wiki.” Accessed: Oct. 22, 2023. [Online]. Available: https://wiki.openstreetmap.org/wiki/Way

[39] “Relation - OpenStreetMap Wiki.” Accessed: Oct. 22, 2023. [Online]. Available: https://wiki.openstreetmap.org/wiki/Relation

[40] A. Diaz-Diaz, M. Ocana, A. Llamazares, C. Gomez-Huelamo, P. Revenga, and L. M. Bergasa, “HD maps: Exploiting OpenDRIVE potential for Path Planning and Map Monitoring,” in *2022 IEEE Intelligent Vehicles Symposium (IV)*, Aachen, Germany: IEEE, Jun. 2022, pp. 1211–1217. doi: 10.1109/IV51971.2022.9827297.

[41] “NVIDIA DRIVE Sim,” NVIDIA Developer. Accessed: Nov. 19, 2023. [Online]. Available: https://developer.nvidia.com/drive/simulation

[42] “HERE HD Live Map | Autonomous Driving System | Platform | HERE.” Accessed: Nov. 19, 2023. [Online]. Available: https://www.here.com/platform/HD-live-map

[43] “Google Maps Platform,” Google for Developers. Accessed: Nov. 19, 2023. [Online]. Available: https://developers.google.com/maps

[44] N. Earth Science Data Systems, “NASADEM: Creating a New NASA Digital Elevation Model and Associated Products | Earthdata.” Accessed: Oct. 22, 2023. [Online]. Available: https://www.earthdata.nasa.gov/esds/competitive-programs/measures/nasadem

[45] “About | OpenTopography.” Accessed: Nov. 19, 2023. [Online]. Available: https://opentopography.org/about

[46] “Getting Started | OpenTopography.” Accessed: Oct. 22, 2023. [Online]. Available: https://opentopography.org/start

[47] “OpenTopography - Data Catalog.” Accessed: Oct. 22, 2023. [Online]. Available: https://portal.opentopography.org/dataCatalog

[48] OpenTopography, “Quantifying Channel Change in a Steep Coastal Stream, CA 2022,” 2023, doi: 10.5069/G9TD9VJF.

[49] OpenTopography, “Copernicus GLO-90 Digital Surface Model,” 2021, doi: 10.5069/G9028PQB.

[50] S.-H. Han, J.-D. Lee, and H.-B. Ahn, “Geospatial data Acquisition Using the Google Map API,” *Int. J. Contents*, vol. 8, no. 1, pp. 55–60, Mar. 2012, doi: 10.5392/IJoC.2012.8.1.055.

[51] “Elevation API Usage and Billing | Google for Developers.” Accessed: Nov. 19, 2023. [Online]. Available: https://developers.google.com/maps/documentation/elevation/usage-and-billing

[52] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, “CARLA: An Open Urban Driving Simulator,” in *Proceedings of the 1st Annual Conference on Robot Learning*, PMLR, Oct. 2017, pp. 1–16. Accessed: Nov. 19, 2023. [Online]. Available: https://proceedings.mlr.press/v78/dosovitskiy17a.html

[53] “SUMO Documentation.” Accessed: Oct. 22, 2023. [Online]. Available: https://sumo.dlr.de/docs/index.html

[54] “RoadRunner Documentation - MathWorks Deutschland.” Accessed: Oct. 22, 2023. [Online]. Available: https://de.mathworks.com/help/roadrunner/index.html?s_tid=srchtitle_site_search_1_RoadRunner&s_tid=mwa_osa_a

[55] “Generate maps with OpenStreetMap - CARLA Simulator.” Accessed: Mar. 03, 2024. [Online]. Available: https://carla.readthedocs.io/en/latest/tuto_G_openstreetmap/

[56] M. Simão, P. Neto, and O. Gibaru, “Taking Advantage of Data Dimensionality Reduction for Dynamic Gesture Recognition from Incomplete Data,” Aug. 2016.

[57] E. W. Weisstein, “Least Squares Fitting.” Accessed: Mar. 03, 2024. [Online]. Available: https://mathworld.wolfram.com/

[58] A. Kaw and J. Paul, “Spline Interpolation Method”.

[59] “Wolfenbüttel topographic map, elevation, terrain,” Topographic maps. Accessed: Mar. 04, 2024. [Online]. Available: https://en-zw.topographic-map.com/map-p6pdn/Wolfenb%C3%BCttel/

[60] “Online OpenDRIVE Viewer.” Accessed: Mar. 03, 2024. [Online]. Available: https://odrviewer.io/

[61] 14:00-17:00, “ISO 26262-1:2011,” ISO. Accessed: Mar. 04, 2024. [Online]. Available: https://www.iso.org/standard/43464.html

[62] “SAE J3016 automated-driving graphic.” Accessed: Mar. 04, 2024. [Online]. Available: https://www.sae.org/site/news/2019/01/sae-updates-j3016-automated-driving-graphic