Lanelets2
Maps for Autoware

Angelo Mastroberardino
Software Engineer
angelo@parkopedia.com

Parkopedia is the global leader in digital parking services
CONTENTS

01 Maps for self-driving vehicles
02 Industry standards for maps
03 Lanelet2
04 How can Lanelet2 help Autoware
1. Maps for self-driving vehicles
Maps for self-driving vehicles
Two approaches to self-driving cars

Robotics Driving Approach

End-to-End Driving Approach
Maps for self-driving vehicles

There are two main approaches to this problem

Robotics Driving Approach

Maps is the **prior knowledge** of the world the car perceives (camera/sensors)
Maps for self-driving vehicles

Cars should drive with their own maps

The map should answer the following needs:

- **Plan a path** from A to B
  - Given the cost / time of each possible path

- **Localize** the car given visual landmarks
  - Artificial
  - Natural

- **Have additional features**
  - Find **parking** in a certain area
  - Visit specific point-of-interests during the planned journey
Maps for self-driving vehicles

What should a map contain?

- **Graph** or roads of a given area
- **Rules** at junctions, stops and traffic lights
- **Point of interests** (parking spaces, …)
- **Shapes** of every lane, or object in the drivable area
  … and the non-drivable (sometimes)
- **Geographical location** of all objects
Maps for self-driving vehicles

Cars should drive with their own maps

As as **service**, a map provider should be

- Up to date with rules and road status
- Precise (less than a meter)
- Light weight
- **Constantly connected**
- Easy to query
Maps for self-driving vehicles
Online or offline maps?

Certain services use **online** tools like Google maps, which heavily rely on GPS signal.

… but that signal is lost in tunnels or indoor spaces.

**Offline** maps guarantees continuity.

… but needs to be uploaded before consumption.

We will talk about offline maps.
We need an indoor map to navigate a car in a parking garage

… and link it to an outdoor map representing roads outside of
2. Industry standards for maps
Industry standards for maps

What maps is the industry using?

1. Navigation Data Standard (NDS)
2. Custom format maps
3. Open Street Map
Industry standards for maps

1. Navigation Data Standard

Data for roads and points-of-interests are saved into an (SQLite) database.

- It is possible to query it in different ways and visualize the contents on a GUI
- It requires a compiler and a subscription to be used

Overview presentation in this tutorial
# Industry standards for maps

## 1. Navigation Data Standard

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Completeness</td>
<td>• High installation cost and time</td>
</tr>
<tr>
<td>• Industry standard for car makers</td>
<td>• Heavy weight</td>
</tr>
<tr>
<td>• It’s a database → easy to query</td>
<td>• Maintenance cost</td>
</tr>
</tbody>
</table>
Industry standards for maps

2. Custom format maps

- Any format (JSON, geo-json, XML, TXT, binary …)
- All development and user tools have to be created in-house
- For example …

```json
{
    "type": "Road 1",
    "name-ref": "Union Street",
    "city": "London",
    "features": {
        "Borders": {
            "geometry": {
                "type": "LineString",
                "coordinates": [
                    [-0.099 ,51.504],
                    [-0.091, 51.503],
                    [-0.092, 51.500],
                    ...
                ],
                "speed-limit": "20mph",
                ...
            }
        }
    }
}
```
Industry standards for maps

2. Custom format maps

Pros

● Implementation with in-house systems
● Freedom to choose the best representation of road objects

Cons

● Non standard format (conversions)
● Requires to solve in-house a few challenges:
  1. Geometry/Location of objects
  2. Traffic rules
  3. Software to view the map
  4. Library to read/write the map
Industry standards for maps

3. Open Street Map

- XML format of maps
- node-way-relation
- classify objects with tags
- contribute maps to the community
- use their maps for free
Industry standards for maps

3. Open Street Map

→ view on youtube
Industry standards for maps

3. Open Street Map

**Pros**
- Free and open source
- Portable
- Has a software to view the map (JOSM)
- Has a library to read/write it, in specific road objects

**Cons**
- Open source project, requires a large community to use it and maintain it
3. Lanelet2
Lanelet2
A library to read and write OSM formatted maps

Created and maintained by FZI Research Center Lanelet2 has mostly 3 functionalities:

1. read/write map efficiently for lanelet formatted Open Street Maps
2. apply rules and connections between roads
3. find the path that connects two lanes

… we will mostly care of the first two
Lanelet2 allowed us to experience several advantages

**Completeness**
- Multi-lane roads
- Traffic rules
- Portable and viewable map (free)

**Flexibility**
- Valid for indoor/outdoor
- Can handle parking spaces
- Overrides
- Easy to tweak for new elements

**Modern software stack**
- C++/Python2/Python3
- Builds with Catkin/ROS

**Data storage**
- Efficient geometry representation
- Library for geometry calculation
A **lanelet** is an atomic section of a lane, which type is identified by a **tag**.

It’s made of a set of geometrical **primitives**

- 2 (directed) *linestrings* → borders and sense
- (optional) 1 middle *linestring* → driving
- (optional) 1 *polygon* → area

The roads type is defined by the tags **subtype** and **location**

And it is related to a set of **regulatory items**
Open Street Maps being in xml format, lanelet exploits the element **tags** to classify objects.

<table>
<thead>
<tr>
<th>type</th>
<th>subtype</th>
<th>location</th>
<th>participant</th>
</tr>
</thead>
<tbody>
<tr>
<td>lanelet</td>
<td>road</td>
<td>urban</td>
<td>vehicle</td>
</tr>
<tr>
<td></td>
<td>road</td>
<td>nonurban</td>
<td>vehicle:car</td>
</tr>
<tr>
<td>bicycle_lane</td>
<td>-</td>
<td></td>
<td>vehicle:car</td>
</tr>
<tr>
<td>crosswalk</td>
<td>-</td>
<td></td>
<td>pedestrian</td>
</tr>
<tr>
<td>stairs</td>
<td>-</td>
<td></td>
<td>pedestrian</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Non road objects in the map can also be identified with one of the built in tags.

For other objects you can **make your own tags** or use **osm tags** like these:

<table>
<thead>
<tr>
<th>type</th>
<th>subtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>parking</td>
</tr>
<tr>
<td></td>
<td>free space</td>
</tr>
<tr>
<td></td>
<td>vegetation</td>
</tr>
<tr>
<td></td>
<td>building</td>
</tr>
<tr>
<td></td>
<td>keepout</td>
</tr>
<tr>
<td></td>
<td>traffic island</td>
</tr>
</tbody>
</table>
A **primitive** is a simple geometry object that constitutes a more complex functional object.

- Point (2d, 3d)
- Linestring (made of points)
- Polygons (closed linestring)

... with a *unique id* → **written once and shared across objects**

... and **const-correctness**: a `ConstLineString3d` will not allow you to modify the underlying Point3d objects that make it
A **regulatory item** is used to express traffic rules. It is an object with an *polygon* area or a *linestring* border, **referenced by** one or more lanelets.

<table>
<thead>
<tr>
<th>type</th>
<th>subtype</th>
</tr>
</thead>
<tbody>
<tr>
<td>regulatory_element</td>
<td>traffic_sign</td>
</tr>
<tr>
<td></td>
<td>traffic_light</td>
</tr>
<tr>
<td></td>
<td>speed_limit</td>
</tr>
<tr>
<td></td>
<td>right_of_way</td>
</tr>
</tbody>
</table>
A point is expressed in Lat/Lng with a tag \texttt{ele} for altitude (2.5D) in the osm file, but in local coordinates in the code.

Lanelet2 \textbf{loads and reprojects} .osm maps to a UTM, but still leaves them in ETRF89.
Let's **read** an open street map

- **Point** (Lat, Lng, Ele)
- **LineString**
- **Polygon**
- **Traffic Rules**
- **Routing Cost**

**Lanelet**

- **Lanelet MAP**
- **IO**
- **Projector**

**Routing Graph**

- **Route**
  - **Lanelet Sequence**
  - **Lanelet Path**
  - **.osm** at origin lat/lng
  - **.osm** from A to B?
  - **.osm** adjacent with lowest cost
  - **.osm** without changing lane

**Lanelet2**

Data flow to read a map
Let’s write an open street map

![Diagram of data flow to write a map]

- **Point** 
  - (Lat, Lng, Ele)
- **LineString**
- **Polygon**
- **Lanelet**
- **Lanelet MAP**
- **Traffic Rules**
- **Routing Cost**
- **IO**
- **Projector**
- **.osm**

**Lanelet2**

Data flow to write a map
Lanelet2
Data flow to write a map

Let's **write** an open street map

- Point
  - (Lat, Lng, Ele)
- LineString
- Polygon
- Traffic Rules
- Routing Cost

Lanelet

Lanelet MAP

Routing Graph

- IO
- Projector

.Route

.osm

Lanelet Sequence without changing lane

from A to B?

write

adjacent with lowest cost

at origin lat/lng

write

adjacent with lowest cost

from A to B?
creating a lanelet map

... programmatically → view on youtube
from OSM to Lanelet2

... tweaking the XML → view on youtube
Lanelet2 demo

Shortest path with lanelet

→ view on youtube
4. How can Lanelet2 help Autoware
How can Lanelet help Autoware

Autoware op_planner needs road data

Currently using vector_map data from csv file
How can Lanelet help Autoware
Lanelet2 could be used to provide data points for the Autoware path-planner

We found three ways to tackle this feature (easiest/worse to hardest/best)

- **Offline**: Save data (osm) -> convert to csv -> vector map loader uses them in op_planner
- **Online**: ROS node sends (osm converted to) vector map data to op_planner
- **Online**: ROS node sends osm data to op_planner (modified to accept them)
How can Lanelet help Autoware

1. Offline: Vectormap converter

This is also part of the PR 2243
How can Lanelet help Autoware

2. Online: ROS node sends (osm converted to) vector map data to op_planner

![Diagram showing the process flow from osm to Lanelet2vectormap, CSV, Autoware2vectormap, to the Open Path Planner](image)
How can Lanelet help Autoware

2. Online: ROS node sends osm data to op_planner

[Diagram showing the flow of data from osm to Autoware2vectormap, with intermediate steps involving lanelet2vectormap and Publish topics, including dtlane, curb, lane, gutter, points, etc.]

osm map server

Open Path Planner
Thank you!