Dynamic vector maps: requirements and uses cases #5409

maxime-clem started this conversation in **Design**

maxime-clem 2 weeks ago Collaborator

Background

A vector map contains detailed information about the environment driven by the ego vehicle. This map is expected to contain information about lanes, crosswalks, traffic lights, speed limits, etc.
Many features of Autoware require a vector map to work properly. For example:

- the planning component where a route, path, and trajectory cannot be generated without a vector map;
- the perception component where the map is used for filtering and generating predictions for the detected objects.

Autoware uses the Lanelet2 library to interact with vector maps.
This library offers many functions to find the optimal route between two points, and and for efficient query of the map.

Motivations for a dynamic vector map

The current Lanelet2 library assumes a single, static source of information. In some cases however, we may want the map to change at runtime.

- We do not want to load one big map but want to load the parts of the map around the ego vehicle (solution proposed here: https://github.com/orgs/autowarefoundation/discussions/4120).
- We want to add new information to the map at runtime (e.g., based on sensor data).
- We want to update the routing costs at runtime (e.g., based on traffic information).

Possible requirements for a dynamic vector map

- Updates to the map can be communicated with messages.
- Updates to the map do not introduce any significant delay.

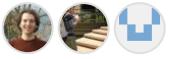
Category

Design

Labels

None yet

3 participants



- Callbacks can be registered for changes to the map (e.g., routing callback if the routing is changed, traffic signals callback if some signals were added/removed, ...).
- Ids used in the map are guaranteed to be consistent between updates.

Please use this discussion to propose other requirements or to present possible use cases for dynamic vector maps.

↑ 3 👍 2

3 comments · 2 replies

Oldest | Newest | Top

YamatoAndo 2 weeks ago Collaborator

@maxime-clem
Thank you for the proposal.
To better understand the content, could you provide a diagram illustrating the architecture you have in mind?

↑ 1 0 replies

yukkysaito last week Maintainer

@maxime-clem
Thank you for your suggestion. Is the background of your proposal to make effective use of online HD map estimation?

↑ 1 1 reply

maxime-clem last week Collaborator Author

This is not really a proposal, just some initial thoughts to start a discussion about dynamic vector maps.
This topic was discussed during the last Planning & Control working group and I just added some rough meeting notes.

maxime-clem last week Collaborator Author

Notes from the discussions during the Planning & Control working group (2024/11/07).

Possible use cases for dynamic lanelet maps:

- Generating map from sensor data (use case from KIT and for https://github.com/orgs/autowarefoundation/discussions/4545).

    - Assuming that the previous version of the map is mostly accurate, we update it with new information.
    - E.g., routing is not changed but lane geometry is updated.

Requirements:

- (for the whole stack) Consistency: small changes to the map can introduce inconsistencies between different modules of the stack.
    - callbacks could be a way to implement that.
    - in Autoware, this issue is maybe solved by having the route calculated only once and communicated to other nodes with the `Route` message.

Possible solutions to keep the map consistent between updates.

- Extending the Lanelet2 library to support updates.
    - main obstacle for dynamic lanelet maps is the routing graph (any small change may require to recalculate the whole routing graph).
    - refactoring of the routing module may be possible: inherit from the current `RoutingGraph`, reuse what you can, but perform fast local dynamic operations.
- Use a global map for routing, use a local, quick to update map for local information.
- Separate format for dynamic maps.

About uncertainty between difference source of information to update the map (sensor, original map file, SD map, ...).

- need metrics for each map element to evaluate how to update the map.
- this problem can be separate from the actual library interface.

About the difference between Lanelet and SD map.

- No "road" in lanelet: in urban context it is difficult to define roads.
- May need to consider that multiple lanelets can be grouped into a higher level entity (e.g, road).
- Tagging lanelets with a corresponding road can be a possible solution.
- Possible interface: SD map is used for routing -> generate sequence of Lanelet ids -> send it to the lanelet library.

Possible architecture:

- One node manages updates to the map and maintain the most up to date map.
- Communicate changes to other consumers of the map.
    - does not publish the whole map but only incremental updates.
- There needs to be one single source of ground truth.

Idea: replicate the lanelet functions with service calls to a central node.

- Main concern: performance impact.
- Good point is that it help abstract away from the map format.

Issue in Autoware:

- Many nodes have their own version of the map in their own memory.
- If the map changes, it needs to change in all nodes (or not ?).

Next steps

- The contributors to the discussion lists their ideas and thoughts.
- Attempt to collect a list of use cases and requirements.
- Try to include more AWF users.

↑ 1 1 reply

maxime-clem 6 hours ago Collaborator Author

Follow up discussion (2024/11/21)

Implement a new routing graph inheriting from the current `RoutingGraph` where the routing graph is not automatically updated but instead must be manually updated.
This would avoid the performance issue with dynamic lanelet maps while keeping the possibility to update the routing graph when needed.

Other possible architecture where the lanelet map can be updated with some messages or API calls.
Motivation: the map is loaded in multiple process/nodes.

- when we have one node publishing updates, should all consumers of the map update their own local version of the map ?
  - in Autoware this makes the most sense.
  - incremental updates may need to be cached so that new consumers can rebuild the up to date map.
    - or one node maintains the up to date map and shares it with any new consumer.
- or one central node maintain the map and share it with every consumer ?
- main concerns are related to performance.
  Managing updates and synchronizing updates between consumers sounds like a common computer science problem where good solutions already exists.
- may be hard to adapt to our use case.

If we can ignore past updates once they are spatially far away, then the central node idea makes sense.

- In some case if we revisit the same place but are missing some past updates, this may yield different versions of the map and introduce inconsistencies.

TODO

- Create an issue with the requirements to initiate work on this task (**@mitsudome-r**).