**System Specification**

Cooperative Research Platform

1. Goal of the document

This document summarizes the details of how the Cooperative Research Platform (CRP) is built up. It consists of the multiple layers:

* System Functionality,
* Architecture,
* Scenario Coverage and demonstration basis

1. Function Specification

This Section describes the high-level specifications of the covered functionality.

Autoware architecture: <https://app.diagrams.net/?lightbox=1#Uhttps%3A%2F%2Fautowarefoundation.github.io%2Fautoware-documentation%2Fmain%2Fdesign%2Fautoware-architecture%2Fnode-diagram%2Foverall-node-diagram-autoware-universe.drawio.svg>

* 1. **Intelligent Speed Adjustment**
* Step 1 functionality: longitudinal speed control adjusted to static information, such as curve and local regulations (speed limit).
* Step 2 functionality: step 1 + speed adjustment on dynamic information, such as moving objects (e.g., followed vehicle).

For both: speed range is 0 <= vx <= 150 kph, which therefore includes automatic start/stop functionality. Function is illustrated in 1. Figure.



1. Figure Function illustration, both step 1 and step 2 functionality

* 1. **Longitudinal Emergency Function**

Functionality: vehicle or delegated sensors provide information about static / dynamic objects. The function decides proper strategy to stop the vehicle (and where to stop it). Then, this strategy is accomplished by applying proper braking force. Function use cases are shown in 2. Figure.

Operation range: .



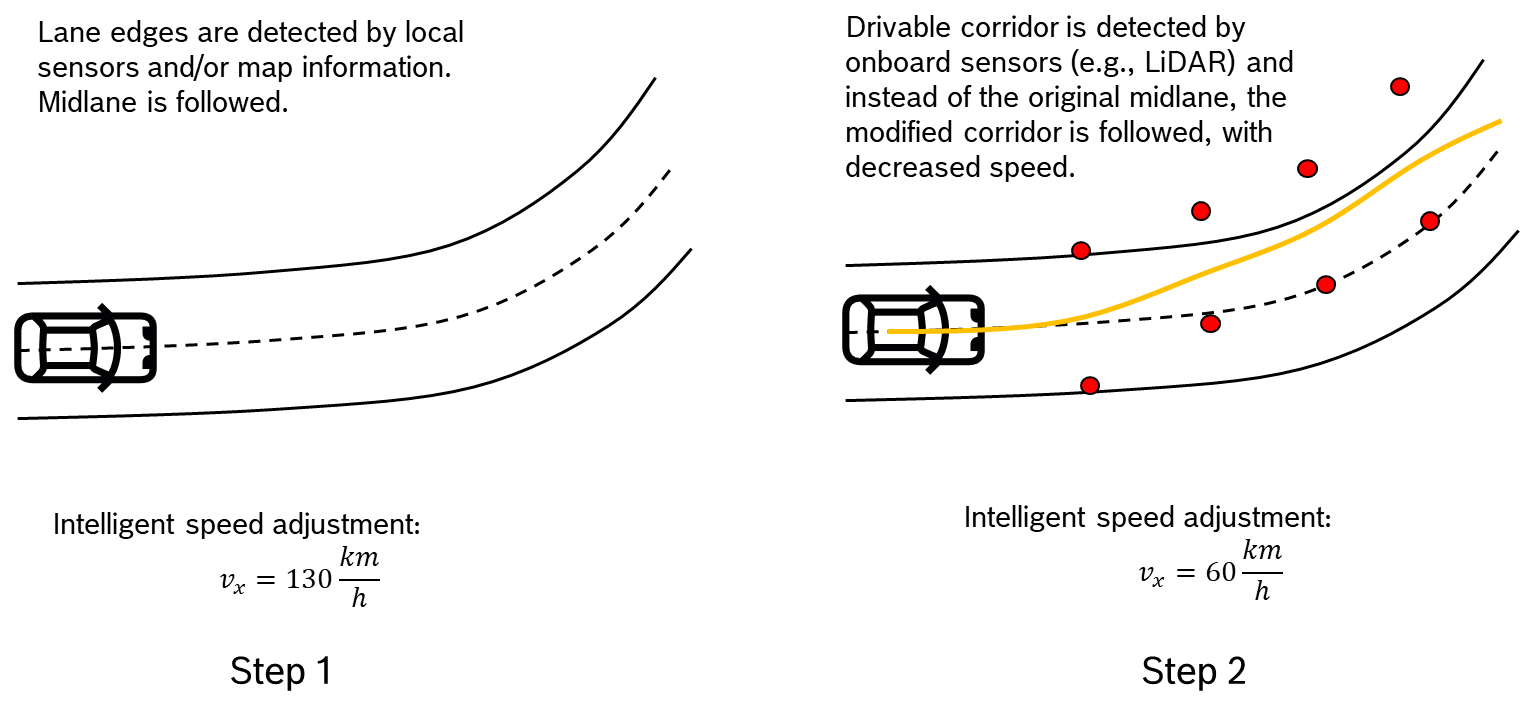
2. Figure Longitudinal emergency use cases from vehicle sensors and infrastructure sensors

* 1. **Lane Follow**
* Step 1 functionality: vehicle is running in a lane, which is bounded by lane edges (markers or only the edge of the drivable surface) and the vehicle follows the centerline of the lane (or externally defined local trajectory).

Operation range: , , .

* Step 2 functionality: Drivable corridor is shifted due to e.g., temporarily shifted road works, which is bounded by 3D obstacles like cones, walls...etc. Vehicle (with lower dynamics) can still navigate through this drivable corridors.

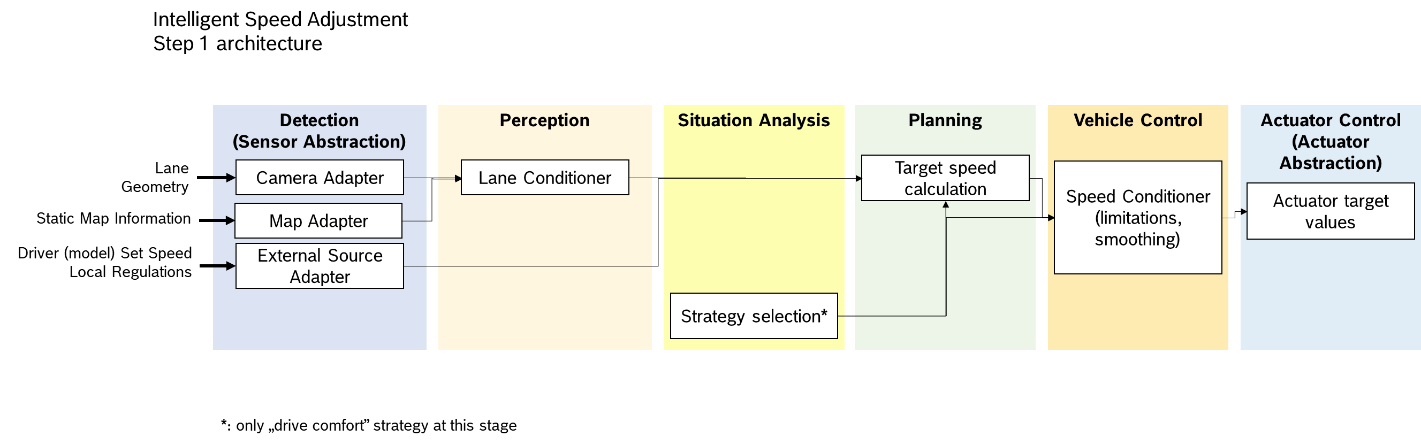
Operation range: , , .



3. Figure Lane follow functionality steps and covered operation.

1. Architecture specification
   1. **Intelligent Speed Adjustment (ISA)**

Step 1 architecture:



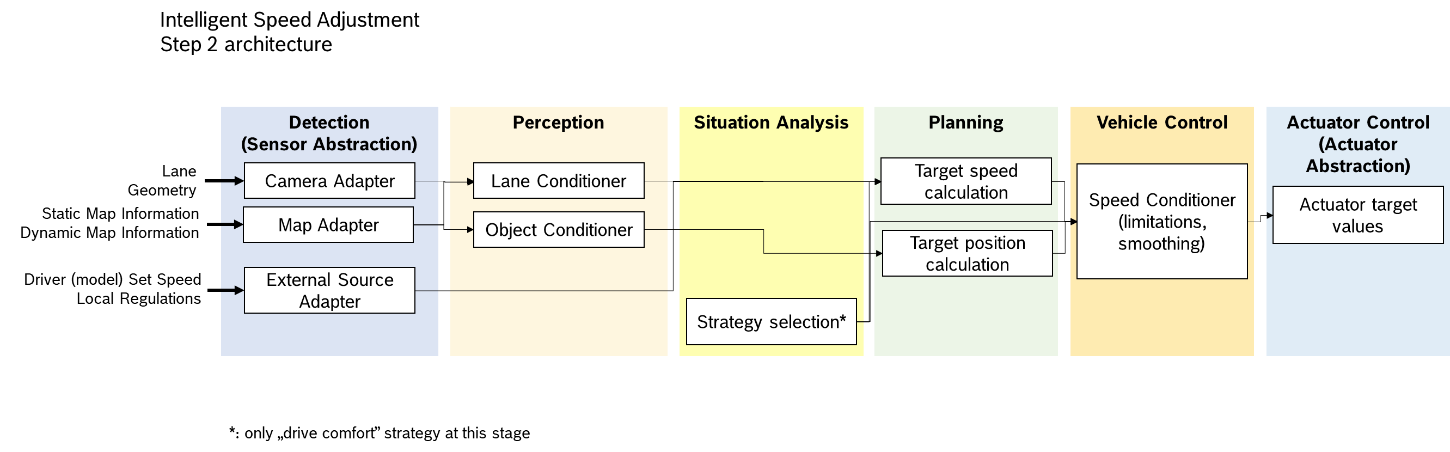
4. Figure. Architecture components of step 1 functionality of ISA

Corresponding Autoware Universe packages:

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Node/Package | Inputs | Outputs |
| **Planning / Target speed calculation** | path\_smoother | behavior\_planning / path; <type: Path > | path\_smoother / path <type: Path > |
| **Planning / Target speed calculation** | obstacle\_velocity\_limiter | motion\_planning / obstacle\_avoidance\_planner / trajectory  <type: Trajectory> | motion\_planning / obstacle\_velocity\_limiter / trajectory  <type: Trajectory> |
| **Planning / Target speed calculation** | motion\_velocity\_smoother | trajectory  <type: Trajectory> | /planning / scenario\_planning / motion\_velocity\_smoother / trajectory  <type: Trajectory> |
| **Detection / MapAdapter** | map\_projection\_loader |  | map\_projector\_info  <type: > |
| **Detection / MapAdapter** | lanelet2\_map\_loader | map\_projector\_info  <type: > | /map  /vector\_map |
| **Vehicle Control / Speed Conditioner** | trajectory\_follower / longitudinal\_controller | /planning / scenario\_planning / trajectory  <type: Trajectory> | /control / trajectory\_follower / control\_cmd  <type: AckermannControlCommand> |

Note: even if we only control the vehicle longitudinally, the lateral path shall be filled with dummy values. Idea: add a straight line with no offset. Later, it must be solved that the vehicle is longitudinally controlled by the system, but laterally by the driver.

Step 2 architecture:

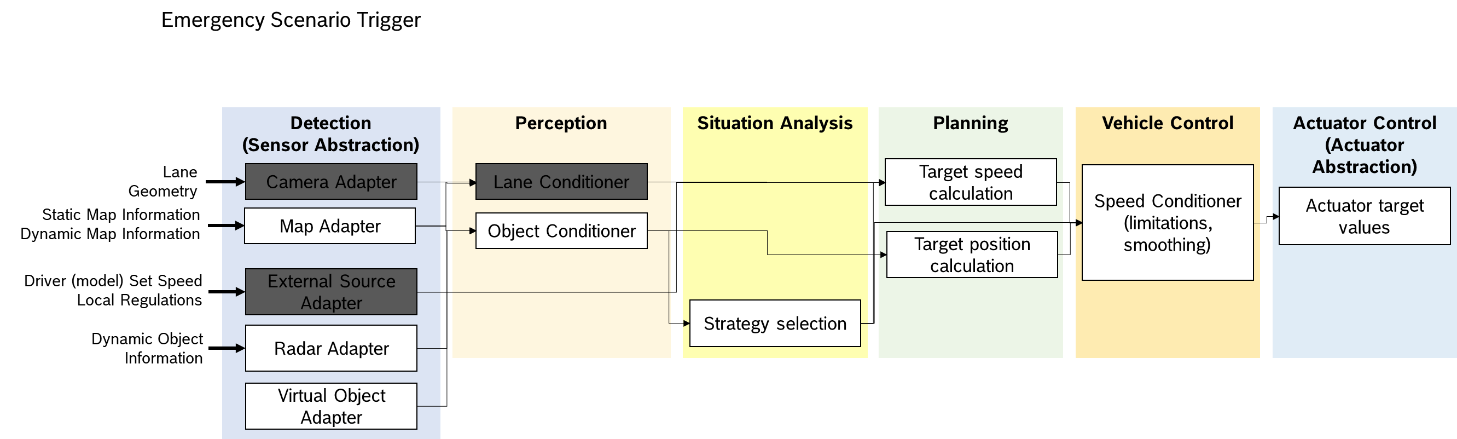


5. Figure Architecture components of step 2 functionality of ISA

Corresponding Autoware Universe packages (above step 1 architecture):

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Node/Package | Inputs | Outputs |
| **Planning / Target position calculation** | autoware\_obstacle\_cruise\_planner | trajectory  <type: Trajectory>  objects  <type: PredictedObjects>  odometry  <type: nav\_msgs::msg::Odometry> | trajectory  <type: Trajectory> |

* 1. **Longitudinal emergency function**



Based on distributed sensor data calculate the trigger of the emergency scenario.

* 1. **Lane follow**

1. Message definitions
   1. **Path**

This is a tier4 autoware message extension, with the following definition:

|  |
| --- |
| std\_msgs/Header header  tier4\_planning\_msgs/PathPoint[] points  nav\_msgs/OccupancyGrid drivable\_area |

* 1. **Path point**

|  |
| --- |
| uint8 REFERENCE=0  uint8 FIXED=1  geometry\_msgs/Pose pose  geometry\_msgs/Twist twist  uint8 type |

* 1. **Trajectory**

|  |
| --- |
| std\_msgs/Header header  tier4\_planning\_msgs/TrajectoryPoint[] points |

* 1. **Trajectory point**

|  |
| --- |
| geometry\_msgs/Pose pose  geometry\_msgs/Twist twist  geometry\_msgs/Accel accel |

1. Autoware packages
   1. **path\_smoother**

<https://autowarefoundation.github.io/autoware.universe/main/planning/autoware_path_smoother/>

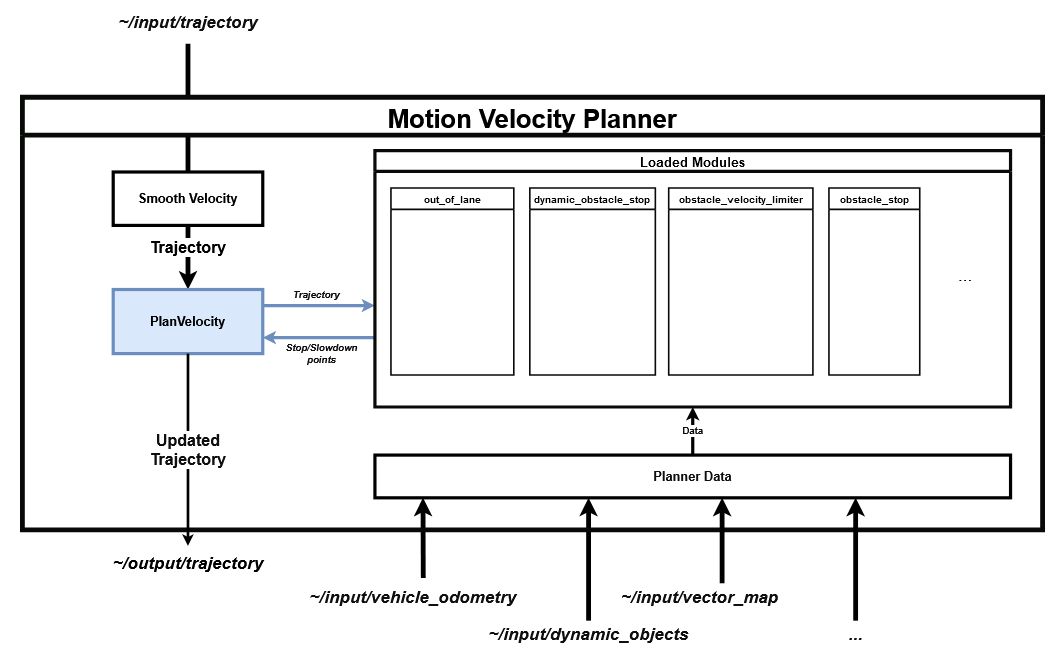
Elastic band concept. This package smooths a path or trajectory.

* 1. **obstacle\_velocity\_limiter**

<https://autowarefoundation.github.io/autoware.universe/main/planning/motion_velocity_planner/autoware_motion_velocity_obstacle_velocity_limiter_module/>

Node limits velocity when driving in the direction of an obstacle. For example, it allows to reduce the velocity when driving close to a guard rail in a curve.

In general it seems, that velocity planning mainly focuses on a) stopping velocities, b) limitation in the obstacle\_velocity\_limiter, when there are obstacles around.



* 1. **motion\_velocity\_smoother**

