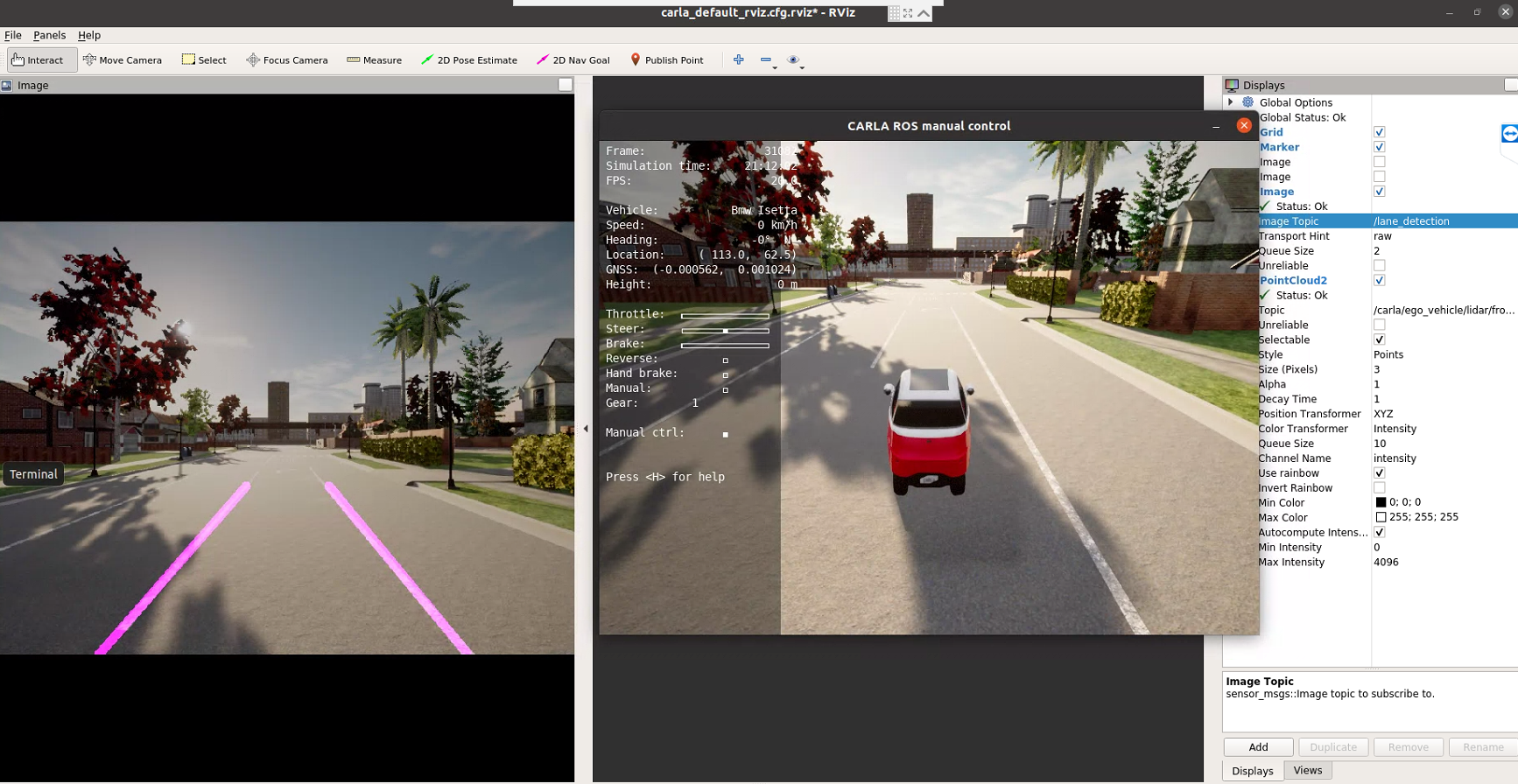
| **DOCUMENT TITLE** | Processes of Automotive Software Development - Lane Detection Feature |
| --- | --- |
| Feature owner | Stevan Stevic |
| Document description | Functional and Nonfunctional requirements for Lane Detection functionality |

# 

# Lane Detection - Project Specification



Goal of this project is to have a software pipeline to identify and track the position of lane lines in front of the vehicle based on raw camera data from CARLA Simulator. Useful features for identifying lane lines are: color, shape, orientation, position and etc.

Basic pipeline can consist of color space transformation, grayscaling, then region masking, then finding edges, and finally using a Hough Transform to identify line segments.

## Requirements

Explanation:

* **shall**: obligatory
* **should**: recommended
* **will**: future requirement
* **may**: desirable, "nice-to-have"

| **ID** | **Description** | **Type** |
| --- | --- | --- |
| LD-1.0 | Project shall be written as a ROS package | Non-functional |
| LD-1.1 | Node shall be able to start the node via launch file | Non-functional |
| LD-1.2 | Project may have unit tests | Non-functional |
| LD-1.3 | Project shall be written in Python | Non-functional |
| LD-2.0 | Node shall be able to receive raw camera frames from CARLA Simulator | Functional |
| LD-2.1 | Node should provide customer with the ability to set a name of the camera topic via launch file (ROS parameters) | Non-functional |
| LD-3.1 | Node shall convert the image to grayscale | Functional |
| LD-3.2 | Node shall blur the image to remove white noise from the image | Functional |
| LD-4.1 | Node shall extract edges using Canny algorithm | Functional |
| LD-4.1 | Node should provide customer with the ability to set Canny parameters via launch file | Non-functional |
| LD-5.1 | Node shall define and mask Region of Interest (ROI) in the image | Functional |
| LD-5.2 | Node should provide customer with the ability to set ROI points as parameters via launch file | Non-functional |
| LD-6.1 | Node shall connect edge pixels into into lines by using Hough transform | Functional |
| LD-6.2 | Node should provide customer with the ability to set Hough transform parameters via launch file | Non-functional |
| LD-7.1 | Node shall extrapolate two lanes from obtained Hough lines | Functional |
| LD-8.1 | Node shall draw two lines on the original image | Functional |
| LD-9.1 | Node shall publish image with drawn lanes to ROS topic | Functional |
| LD-9.2 | Node should provide customer with the ability to set name of topic for publishing image with lanes via launch file. Default: /lane\_detection | Non-functional |

### Guidеlines

Some OpenCV functions (beyond those introduced in the lesson) that might be useful for this project are:

* cv2.inRange() for color selection
* cv2.fillPoly() for regions selection
* cv2.line() to draw lines on an image given endpoints
* cv2.addWeighted() to coadd / overlay two images
* cv2.cvtColor() to grayscale or change color
* cv2.imwrite() to output images to file
* cv2.bitwise\_and() to apply a mask to an image

Visualize the results using Rviz configuration **in carla-autoware docker**:

rosrun rviz rviz -d /home/autoware/carla\_ws/src/ros-bridge/carla\_ros\_bridge/config/carla\_default\_rviz.cfg.rviz

Lidar detection point cloud messages:  
**type**: *sensor\_msgs::PointCloud2***topic**: “/carla/ego\_vehicle/lidar/lidar1/point\_cloud”

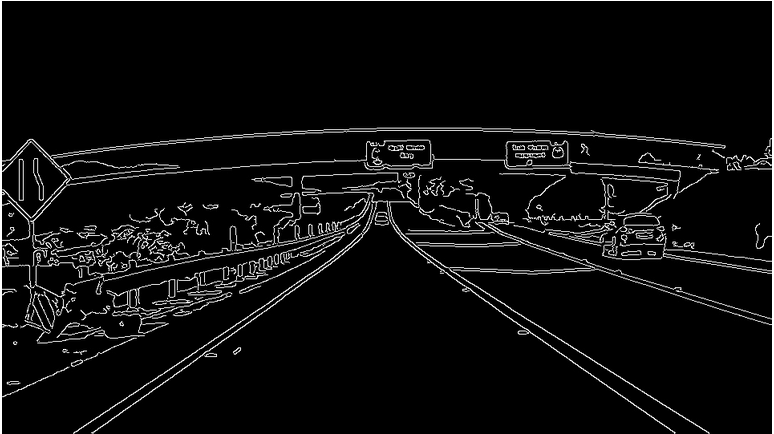
RGB Image raw  
**type**: sensor\_msgs/Image  
**topic :** /carla/ego\_vehicle/camera/rgb/front/image\_color

Useful links:

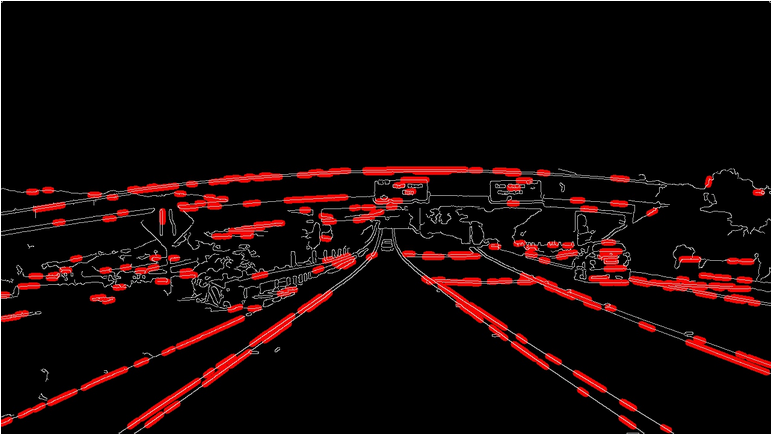
* Python ROS package structure - <http://www.artificialhumancompanions.com/structure-python-based-ros-package/>
* ROS Tutorial - <http://wiki.ros.org/ROS/Tutorials>
* Hough transform explanation - <https://alyssaq.github.io/2014/understanding-hough-transform/>

Pipeline step visualizations:

* Canny edge detection



* Hough Transform



* Region masking

