Autonomous Parking Spot Locator and Park System for Empty Parking Lot

April 20, 2016

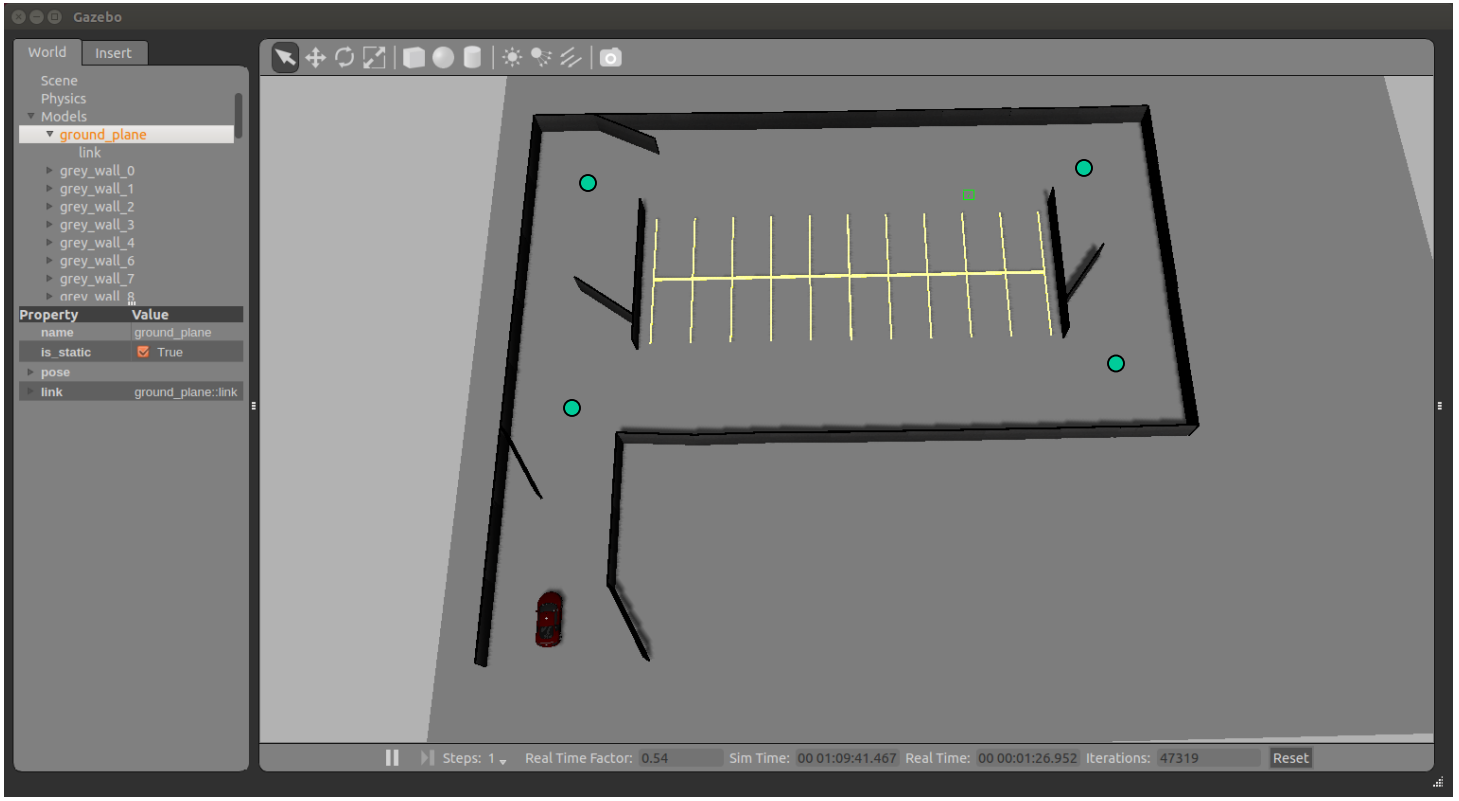
Kazumi Malhan

Environment Setup & Challenges

Parking Lot Layout

Known
Initial heading
4 GPS points

Gazebo Issue

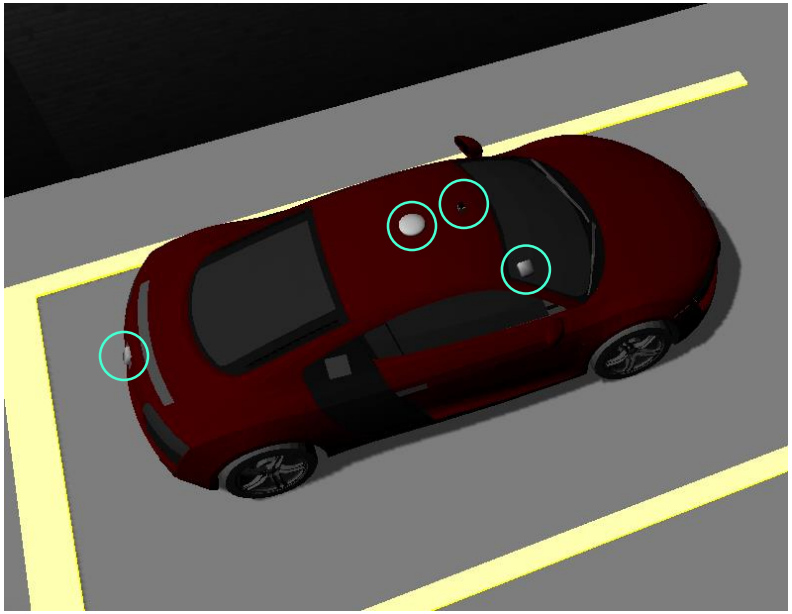


Challenges

- Avoid obstacles during the navigation time.
- Find parking lot using camera.
- Fix movement if car is going out of lane during parking.

Vehicle Configuration

Audi R8 (Audibot)



Sensor Configuration

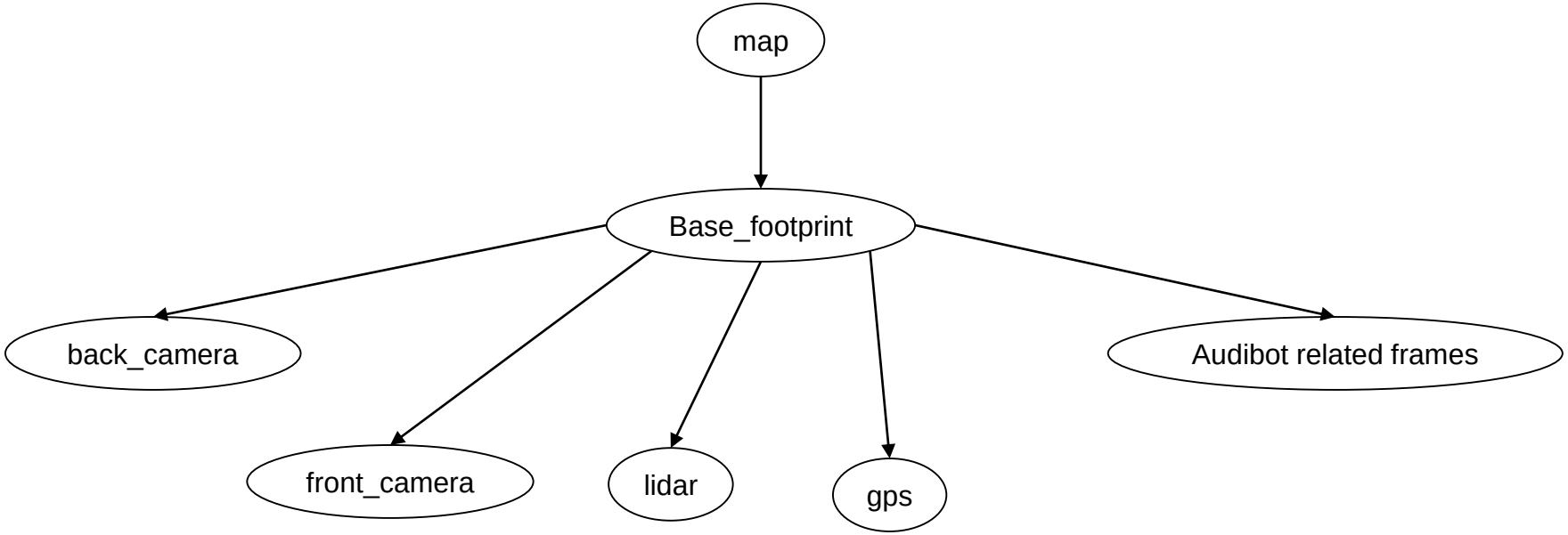
Sensors	Purpose
GPS (1Hz)	Localization
Lidar	Cost map Generation
Front Camera	Parking Lot Detection
Back Camera	Parking Movement Fix

- Heading is not published.
- Initial heading is known.

Vehicle Control

- Vehicle is controlled using speed (m/s) and steering angle.
- No manual control during entire process.
- Vehicle publishes current speed, yaw rate, and steering angle.

TF Frames



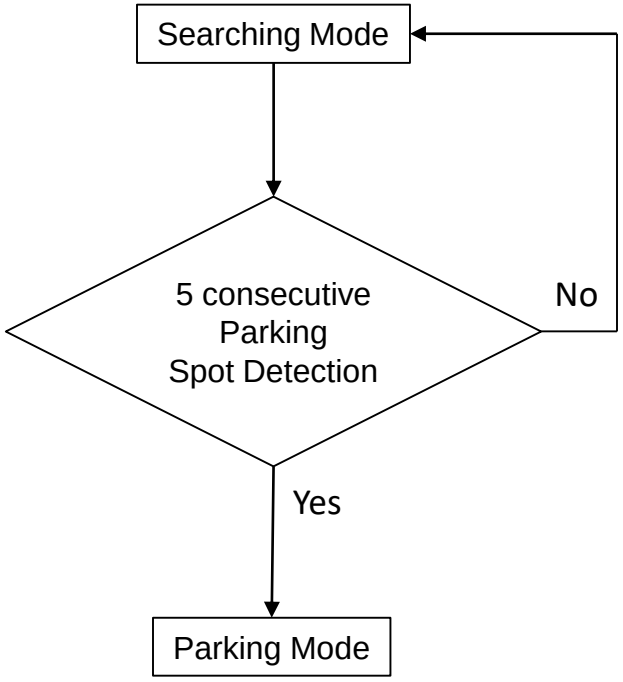
- Map frame to base_footprint frame transform is broadcasted by odom_node.
- World frame is not published.
- Other transforms are done by Gazebo.

Integration (1) – Control Selector

The project consists of three major components

1. Navigation
2. Locate Parking Spot
3. Parking Maneuver

Control selector node monitors parking spot detection topic.



Searching Mode

Use navigation node to search through the parking lot by itself.

Parking Mode

Use parking movement node to perform parking maneuver.

Implementing Bicycle State Space Model

$$State\ Vector \qquad X = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}$$

$$Control\ Vector \qquad U = \begin{bmatrix} v \\ \alpha_s \end{bmatrix}$$

$$State\ Function \qquad f(X,U) = \begin{bmatrix} \dot{x} = v cos\psi \\ \dot{y} = v sin\psi \\ \dot{\psi} = \frac{v}{L}\tan\left(\frac{\alpha_s}{\gamma}\right) \end{bmatrix}$$

Incorporate GPS

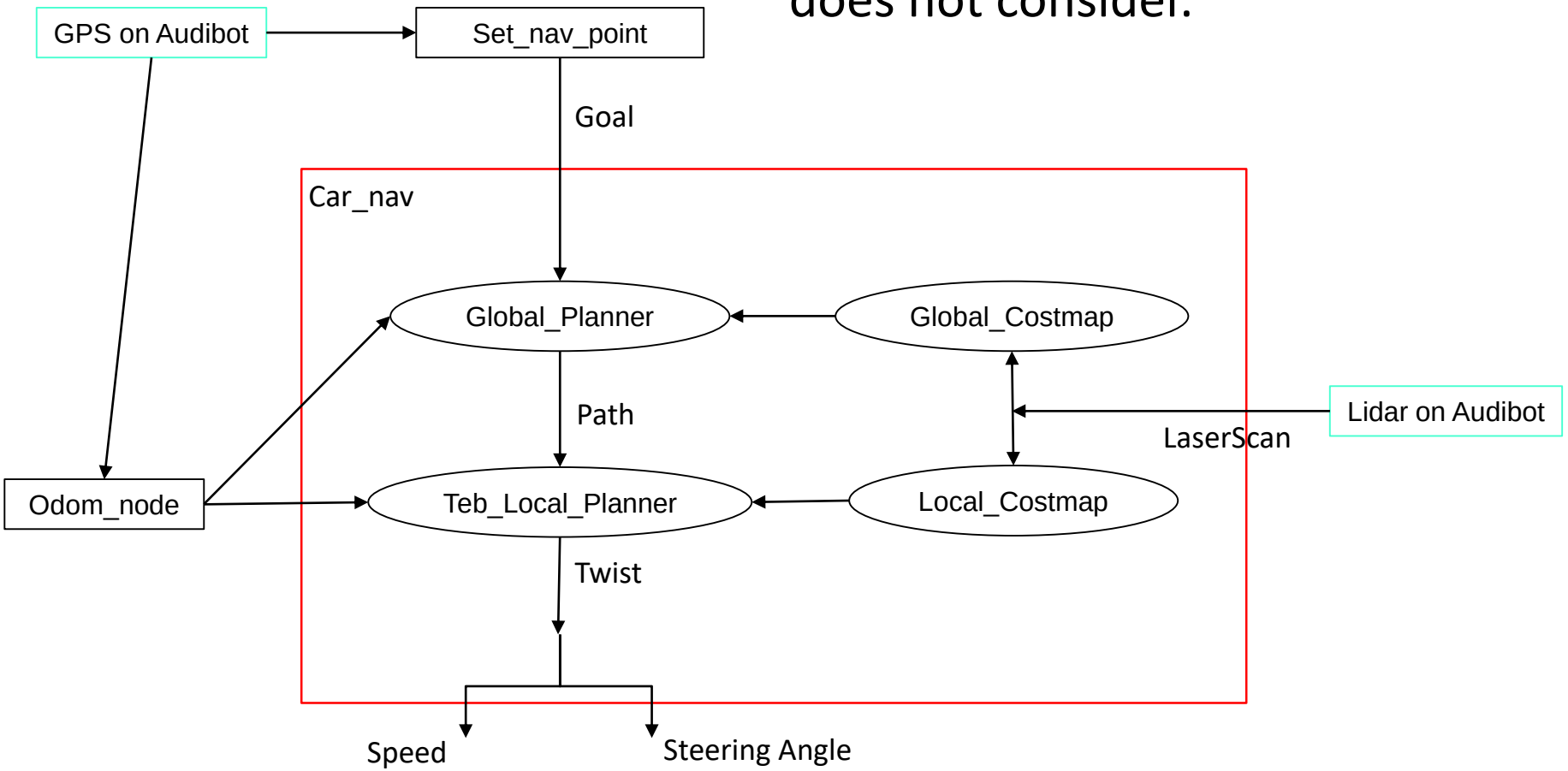
- System is running at 50 Hz.
- GPS is running at 1 Hz.
- System replaces x, y with GPS value when GPS is available.

State Space Discretization

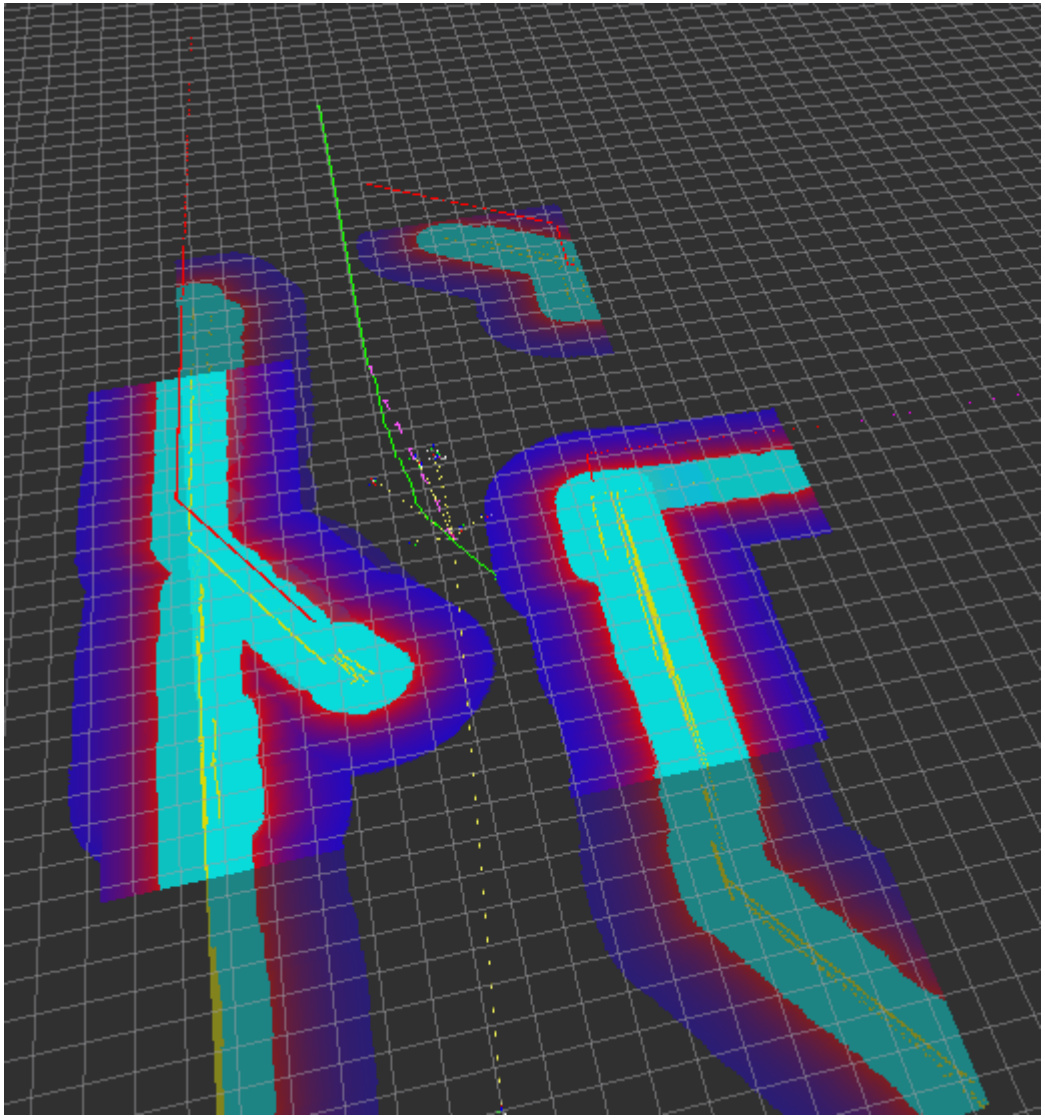
$$x_{k+1} = x_k + T_s * v_k * cos\psi_k$$
$$y_{k+1} = y_k + T_s * v_k * sin\psi_k$$
$$\psi_{k+1} = \psi_k + T_s * \frac{v_k}{L}\tan\left(\frac{\alpha_{s|k}}{\gamma}\right)$$

Navigation (2) – Cost Map and Path Planning

Teb_local_planner considers carlike movement, but other components does not consider.



Navigation (3) – Rviz Visualization



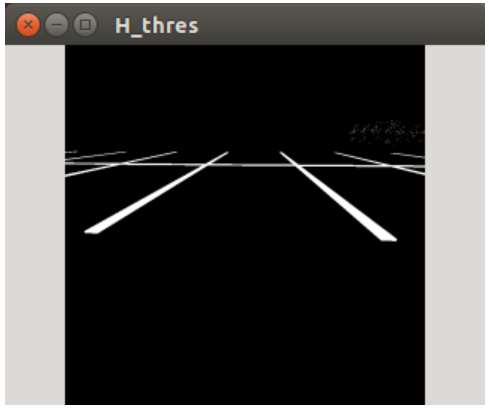
Problems Encountered

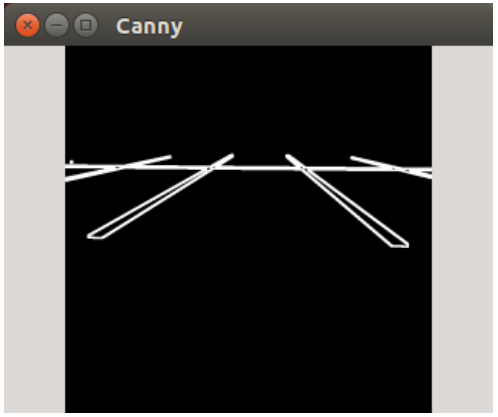
- Global Plan has sharp turn
- Costmap configuration
- Teb_local_planner external libreay dependency

- Green -> Global Plan
- Pink -> Local Plan
- Red -> Laser Scan
- Costmap
  - Global (dark)
  - Local (bright)

Parking Spot Detection (1) – Image Process

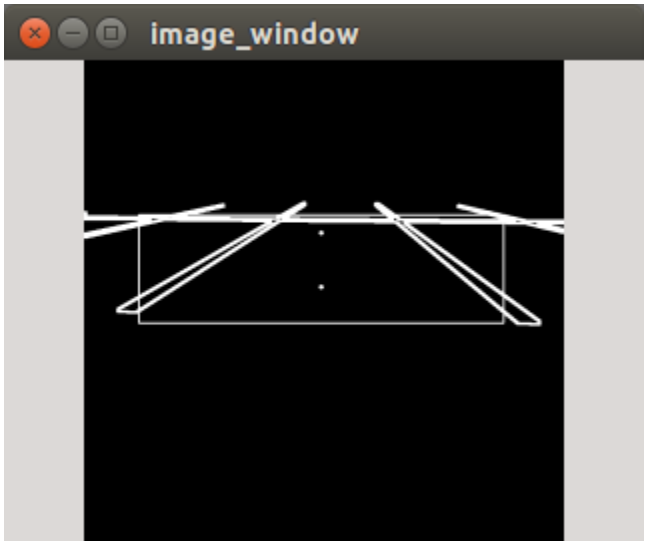
1. Convert image from BGR8 to HSV
2. Split Channel, and take Hue
3. Threshold Hue (center: 30, width: 10)
4. Erode and dilate to remove noise
5. Apply Canny Edge Detection algorithm
6. Erode and dilate to remote noise
7. Apply Template Matching, get highest possible location
8. Inside the location, apply HoughP algorithm.
9. Calculate angles of each line, and delete similar ones.
10. Check angle and number of lines detected





Parking Spot Detection (2) – Template Matching

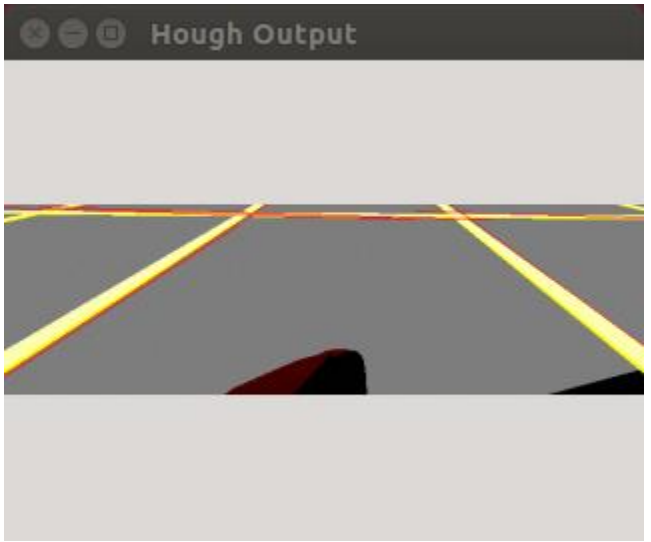
Step1: Compares a template against overlapped image regions.
Step2: Finds the global minimum in the array
Step3: Extract regions



Template

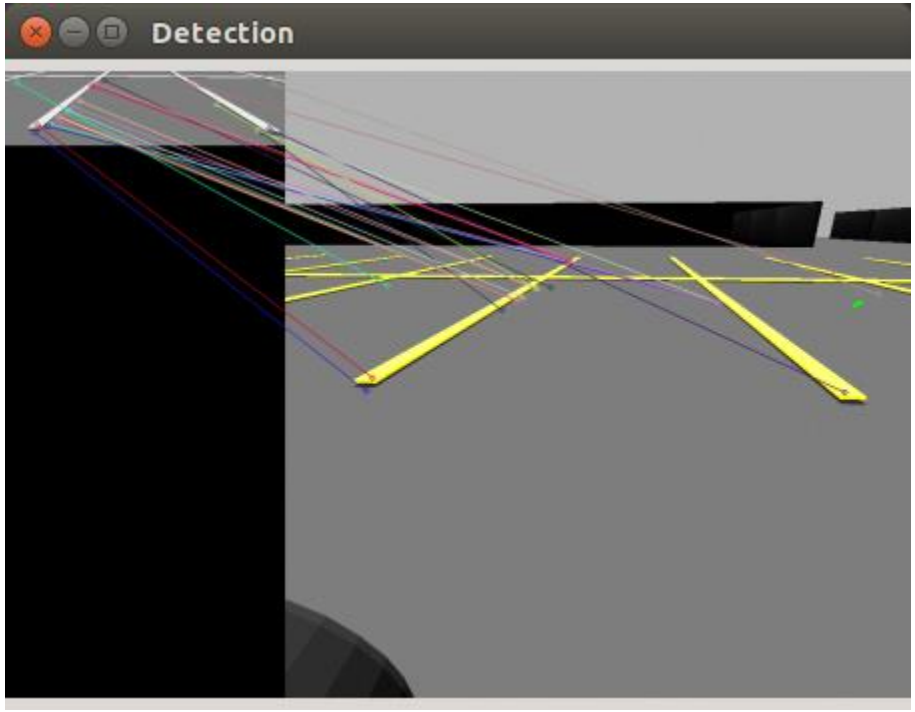
HoughP algorithm

Detect two near vertical lines and one horizontal line.



Parking Spot Detection (3) – Feature2d

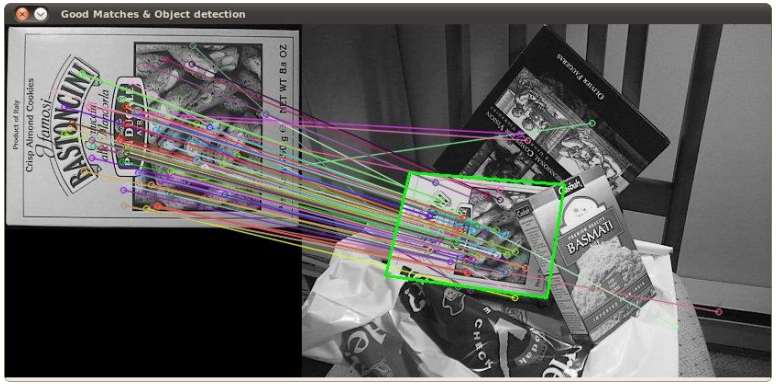
Feature2d is another method to detect objects in the image.
Uses key point matching.



This method did not work well

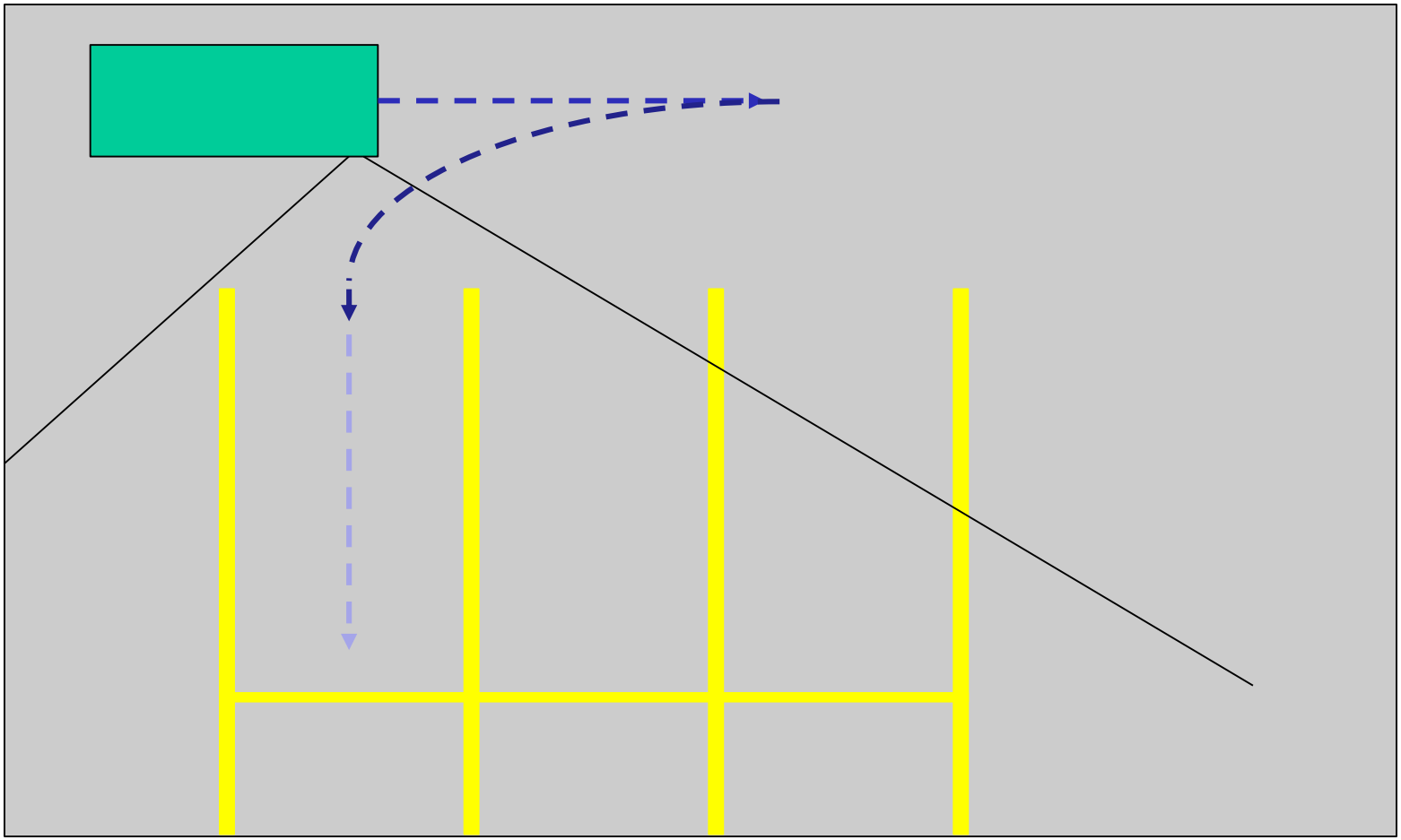
- Less unique feature in parking lot lines.
- Parking lot is repeat of lines.

Sample working image (from OpenCV website)



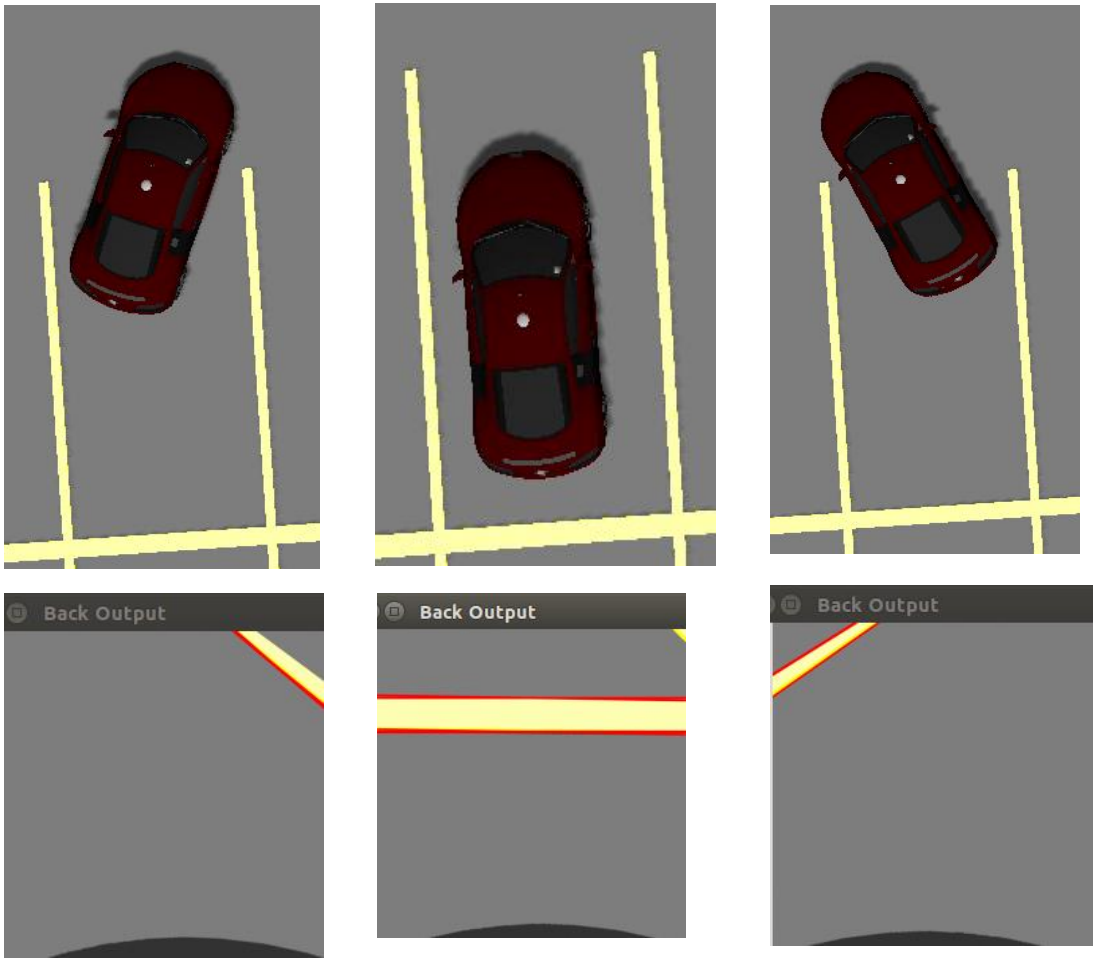
Parking Maneuver (1) – Basic Control

Basic parking maneuver is pre-computed based on vehicle dynamics, camera mount position.



Parking Maneuver (2) – Redo Process

Basic controller gets near the parking spot, but need precise control. Feedback is provided to control node via back_camera image.



Left Center Right

Same steps till canny is applied to back camera.

Hough Line to identify lines, delete similar ones.

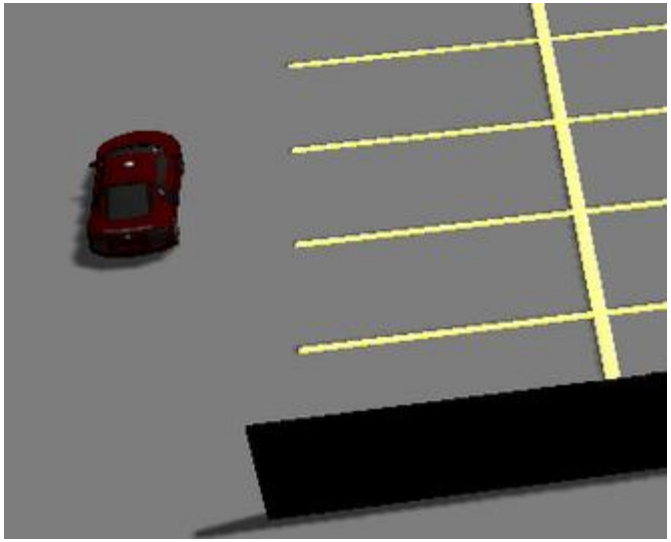
Based on line's rho (distance) and theta (angle), classifies status.

Image Transport – Better Parking Maneuver

Effort to transport Image to map frame

1. Each pixel to field of view (FOV)
2. FOV to Camera frame
3. Camera frame to Base_footprint frame
4. Base_footprint frame to map frame

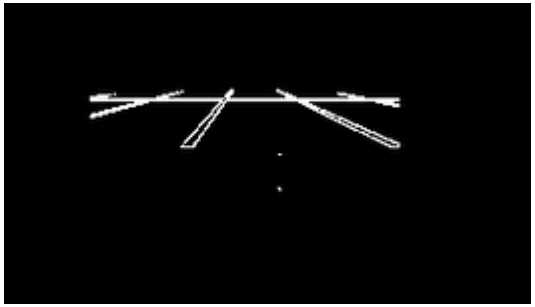
Gazebo World



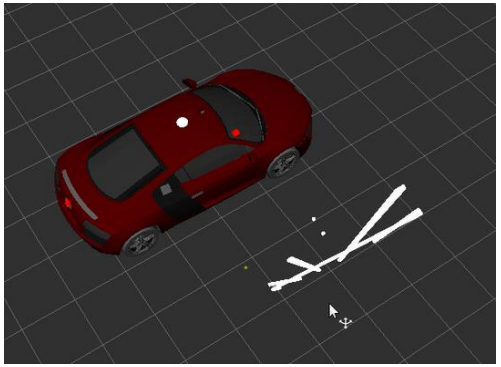
Did not fully implemented due to lack of development time.

This allows system to evaluate better parking maneuver. (Future Task)

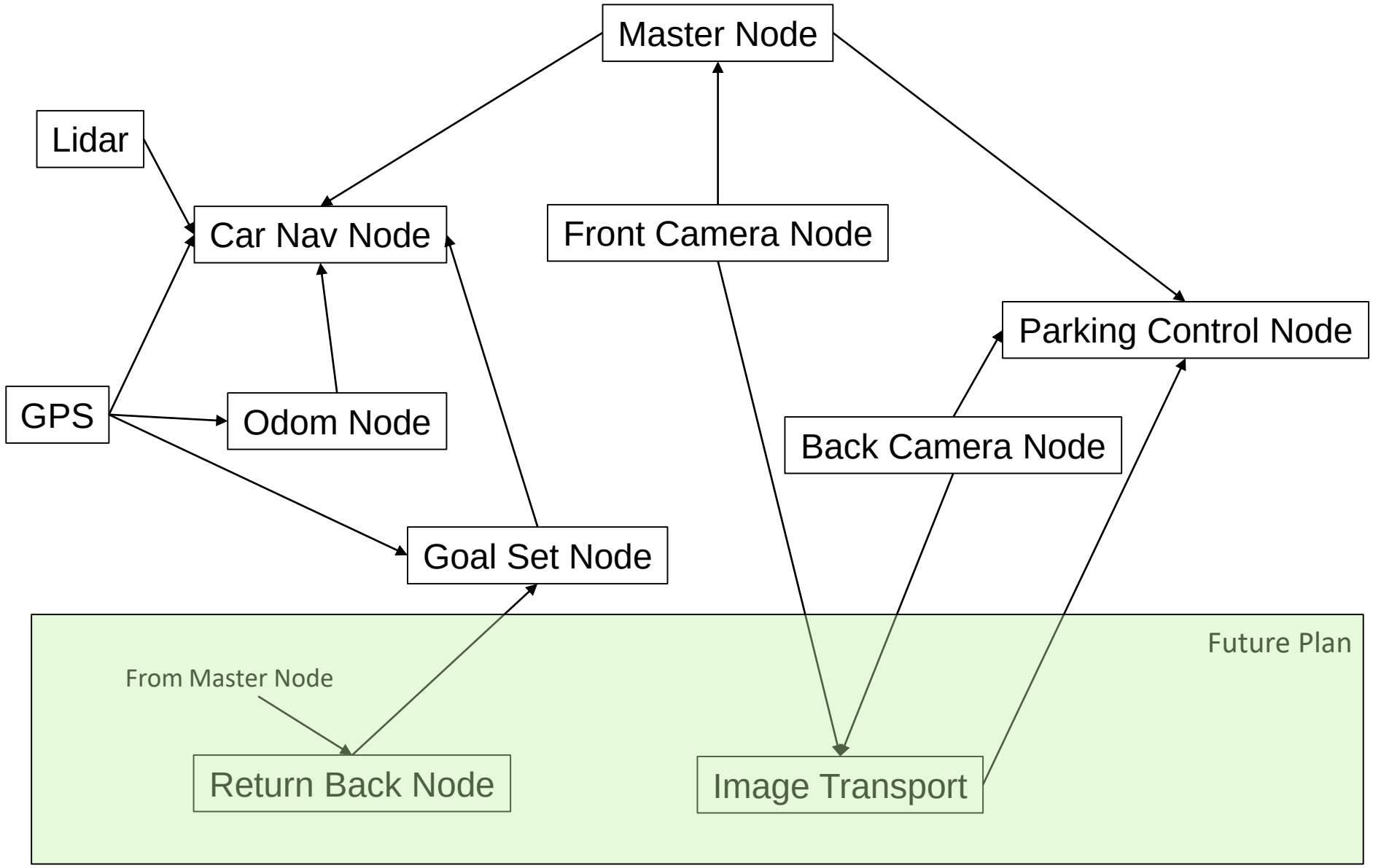
Image



Rviz



Integration (2) – Node Implementation



Conclusion

- Able to implement each component (navigation, parking spot detection, and parking maneuver), and integrate all together.
- Got chance to learn about image processing and OpenCV library.
- Learned the difficulty of converting sensor inputs to useful controller outputs.

Future Plan

- Fully implement “Image Transport” node for better parking trajectory planning.
- Feature to record GPS points during searching, so car can return to original location when requested.

Thank You Very Much