Polaris GEM e2 Vehicle

Hang Cui
Feb. 10, 2022
Contents

**Hardware & Software** of Polaris GEM e2 Vehicle

**Simulator** of Polaris GEM e2 Vehicle

**Lane Tracking**

Sensor/Actuator Calibration for Lane Detection and Tracking

GPS-based way points tracker

Path Tracking with Pure Pursuit Controller and Single Monovision Camera
Hardware of Polaris GEM e2 Vehicle

Velodyne VLP-16 LiDAR (Top LiDAR)
Mako G319C Camera (Front Camera)
Delphi ESR 2.5 Radar (Front Radar)
ProPak6 & SPAN-IGM-S1 (GNSS/INS)
PACMod (DBW Kit)
AStuff Spectra 2 (2x RTX 2080)
Hardware of Polaris GEM e2 Vehicle
Software of Polaris GEM e2 Vehicle

**System:**

**Ubuntu 20.04 with ROS Noetic (Python 3) and Gazebo 11**

NVIDIA Driver Version: 460.80
Qt 5.12.11
CUDA 11.0.3
cuDNN 8.1.1
TensorRT 8.0.0.3
OpenCV 4.4.0
pytorch 1.7.1
tensorflow 2.5
PCL 1.11.1
Simulator of Polaris GEM e2 Vehicle

The GEM simulator was originally built for ROS Melodic / Gazebo 9 (Ubuntu 18.04).

It has been **improved** and merged into **ROS Noetic / Gazebo 11 (Ubuntu 20.04)**.
Simulator of Polaris GEM e2 Vehicle

The GEM simulator was originally built for ROS Melodic / Gazebo 9 (Ubuntu 18.04)

It has been improved and merged into ROS Noetic / Gazebo 11 (Ubuntu 20.04)
Simulator of Polaris GEM e2 Vehicle

The GEM simulator was originally built for ROS Melodic / Gazebo 9 (Ubuntu 18.04)

It has been **improved** and merged into **ROS Noetic / Gazebo 11 (Ubuntu 20.04)**
Simulator of Polaris GEM e2 Vehicle

Updates & New Features:

(1) Internal collision bugs were fixed.

(2) Model’s inertia, friction and damping were tuned. (*may still need more tunings*)

(3) A frontal 2D laser was added.

(4) Optionally spawn LiDAR or laser sensor by using ROS launch commands.

(5) Model’s domain has been unified, such as /gem/imu/..., /gem/gps/..., /gem/...

(6) Model’s base_link was redefined at the center of the rear axle.

(7) All sensor frames were calibrated/measured with respect to the base_link.

(8) Optionally spawn multiple GEMs into the environment (*working on it*)

(9) A `polaris_gem_drivers_sim` is under development (`gem_gps_tracker_sim`, `gem_vision_tracker_sim`, `gem_teleop`, `gem_pure_pursuit_sim`, etc) to work with this simulator, namely `polaris_gem_simulator`

(10) Coming more
Simulator of Polaris GEM e2 Vehicle

The high bay environment in real size with matched GPS coordinates.

Lat: 40.09302494
Lon: -88.23575484
x: -5.5
y: -21
Simulator of Polaris GEM e2 Vehicle

Lat: 40.09302494
Lon: -88.23575484
Simulator of Polaris GEM e2 Vehicle

Lat: 40.09302494
Lon: -88.23575484
Simulator of Polaris GEM e2 Vehicle
Simulator of Polaris GEM e2 Vehicle
Steering Wheel to Front Wheel Calibration

\[ \delta_s = -0.1084 \delta_f^2 + 21.755 \delta_f \]

\[ \delta_f = \arctan\left(\frac{2}{\tan \delta_l + \tan \delta_r}\right) \]
Steering Wheel to Front Wheel Mapping
Steering Wheel to Front Wheel Calibration

<table>
<thead>
<tr>
<th>Left Wheel</th>
<th>Right Wheel</th>
<th>Middle Angle</th>
<th>Steering Angle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4.6</td>
<td>4.2</td>
<td>4.4</td>
<td>90</td>
</tr>
<tr>
<td>10.5</td>
<td>8.5</td>
<td>9.4</td>
<td>180</td>
</tr>
<tr>
<td>14</td>
<td>12</td>
<td>12.9</td>
<td>270</td>
</tr>
<tr>
<td>18.8</td>
<td>17.3</td>
<td>18</td>
<td>360</td>
</tr>
<tr>
<td>24.2</td>
<td>22.4</td>
<td>23.3</td>
<td>450</td>
</tr>
<tr>
<td>31.7</td>
<td>26.5</td>
<td>28.9</td>
<td>540</td>
</tr>
<tr>
<td>38</td>
<td>33</td>
<td>35.3</td>
<td>630</td>
</tr>
</tbody>
</table>

Steering wheel was controlled by program to rotate at a particular angle, namely 0, 90, 180, 270, 360, ..., etc.
Lane Detection and Tracking - Bicycle Model
Lane Detection and Tracking - Pure Pursuit Controller

\[ \dot{x} = v \cos \theta \]
\[ \dot{y} = v \sin \theta \]
\[ \dot{\theta} = \frac{v}{L} \tan \delta_f \]
\[ \dot{v} = a \]

\[ \delta_f = \tan^{-1} \left( \frac{2L \sin \alpha}{l_c} \right) \]
Lane Detection and Tracking - Stanley Controller

\[
\delta_e = \tan^{-1}\left(\frac{ke}{v}\right)
\]

\[
\theta_{ref} = \theta_{ref} - \theta
\]

\[
\delta_f = \theta_e + \delta_e
\]

\[
e = [\sin\theta_{ref}(x-x_{ref}) - \cos\theta_{ref}(y-y_{ref})]
\]

$L$: wheelbase
$e$: cross track error
$\theta_e$: heading error
$\delta_e$: cross track steering
Lane Tracking with Pure Pursuit Controller

Autoware.AI

Curve the vehicle follows

Target point

Self-position
Reference

ASTuff drivers: https://github.com/astuff

Polaris GEM e2 ROS Noetic simulator: https://github.com/hangcui1201/POLARIS_GEM_e2

Polaris GEM e2 user manual: https://github.com/hangcui1201/POLARIS_GEM_e2
Thanks!