Dataspeed Drive-by-Wire System Gazebo/ROS Simulator

Point of Contact:
(support@dataspeedinc.com)

Version: 2.6.2

Contents

SIMULATOR VERSION HISTORY

Version 2.6.2 (February 2022)

- Minor updates to match upstream DBW ROS driver changes.

Version 2.6.1 (February 2022)

- Fixed a bug in Pacifica brake command message parsing

Version 2.6.0 (December 2021)

- Extracted demo simulation configurations into a separate package in the public repository.
- Modified lane keeping detection and control nodes to track all lines in the image instead of the two closest ones.
- Removed model year distinction from simulation parameters.
- Added support for turn signal commands and added a topic to simulate the physical state of the turn signal lever.
- Changes to example GPS receiver:
  - Renamed UTM odometry output topic from 'utm' to 'odom'
  - Added a NMEA string output topic called 'gga'
  - Added a `ref_alt` SDF parameter, and the NavSatFix output reports altitude

Version 2.5.0 (August 2020)

- Adds support for Ubuntu 20.04 and ROS Noetic
- Breaks support for Ubuntu 16.04 / ROS Kinetic and Ubuntu 18.04 / ROS Melodic

Version 2.4.0 (August 2020)

- Minor bugfixes
- Changed URDF joint names
  - steer_fl → steer_fl_joint
  - steer_fr → steer_fr_joint
  - wheel_fl → wheel_fl_joint
  - wheel_fr → wheel_fr_joint
  - wheel_rl → wheel_rl_joint
  - wheel_rr → wheel_rr_joint

- Changed the UTM odometry message output from the PerfectGps plugin
  - Velocity now represented in vehicle frame instead of UTM frame
  - Frame ID in the header reflects the current UTM zone

Version 2.3.0 (November 2019)

- Added simulation of Ford F-150.
- Improved braking model for Jeep Grand Cherokee.
- Changed launch file arguments for the example simulations.

Version 2.2.0 (October 2019)

- Added simulation of Jeep Grand Cherokee.

Version 2.1.2 (August 2019)

- Example GPS sensor plugin populates UTM odometry message with the correct child frame ID after the addition of the TF prefix option in v2.1.0.
- Documented the 'enhanced_fix' topic published from the example GPS sensor

Version 2.1.1 (July 2019)

- Added pub_odom YAML configuration parameter

Version 2.1.0 (June 2019)

- Separated vehicle Gazebo model name from its TF prefix and ROS topic namespace
- Gave user the option to not use a TF prefix

Version 2.0.0 (March 2019)

- Dropped support for ROS Indigo / Gazebo 2.x, added support for ROS Melodic / Gazebo 9.x
- Added simulated interface to the new Universal Lat/Lon Controller (ULC) drive-by-wire system feature
- Added support for the Pacifica platform
- Fixed desync between camera control plugin and Gazebo GUI updates
- Added a first-person mode to the camera control plugin
- Exposed reference coordinates for the production GPS receiver as YAML parameters

Version 1.2.0 (January 2018)

- Updated multi_robot_sim to demonstrate TF publishing and Velodyne point clouds.
- Simulator reports the production GPS output messages over the simulated CAN bus.
- Minor bug fixes.

Version 1.1.0 (October 2017)

- Added a simulated Velodyne VLP-16 LIDAR to the default URDF file.
- Implemented a much more true-to-life steering control dynamics model.

Version 1.0.5 (June 2017)

- Initial release.

1 Installation

Table 1 shows which Ubuntu and ROS distributions are compatible with the 2.x.x versions of the drive-by-wire system simulator.

	2.0.x	**2.1.x**	**2.2.x**	**2.3.x**	**2.4.x**	**2.5.x**	**2.6.x**
Ubuntu 20.04 / ROS Noetic						▨	▨
Ubuntu 18.04 / ROS Melodic	▨	▨	▨	▨	▨		▨
Ubuntu 16.04 / ROS Kinetic	▨	▨	▨	▨	▨		

Table 1: Ubuntu and ROS support for each minor version update.

To install the latest Dataspeed drive-by-wire system simulator that is released for your installed ROS distribution, you will first need to set up your computer to accept software from the Dataspeed package server.

To do this, follow the **Setup apt** and **Setup rosdep** instructions found here:
https://bitbucket.org/DataspeedInc/ros_binaries.
Then install the actual simulator package:

```
sudo apt install ros-$ROS_DISTRO-dataspeed-dbw-simulator
```

2 Core Simulation Components

2.1 Installed Binary Packages

After installing the simulator, the following packages are installed in `/opt/ros/$(ROS_DISTRO)`:

- **dataspeed_dbw_gazebo** – 3D model assets, launch files, and default configuration files.
- **dataspeed_dbw_gazebo_demo** – Lane detection and trajectory control node binaries used in the lane keeping demo simulation (Section 4.1.3).
- **dataspeed_dbw_gazebo_plugins** – Gazebo plugin binaries that implement the behavior of the emulated CAN interface to the Dataspeed drive-by-wire system.
- **dataspeed_dbw_simulator** – ROS Metapackage

2.2 Launch File to Start the Simulator

dataspeed_dbw_gazebo.launch in the **dataspeed_dbw_gazebo** package is the main launch file used to start the simulator. There are four arguments to this launch file:

- **use_camera_control** – This is a boolean argument that enables or disables a Gazebo plugin to automatically move the camera to follow a given model in the world. If unspecified, this argument defaults to true. See Section 2.3 for how the camera control plugin works.
- **world_name** – This is a path to a Gazebo world file to load at startup. If unspecified, an empty world will be used. To save a world file for future simulations, be sure to delete any vehicle models present in the world before saving.
- **sim_param_file** – This is a path to a YAML file containing simulation configuration parameters. If unspecified, the **default_sim_params.yaml** file in the **dataspeed_dbw_gazebo** package is used. See Section 3 for YAML format details and the available configuration options.
- **headless** – This is a boolean argument that enables or disables the Gazebo GUI.

2.3 Camera Control Plugin

The camera control plugin is used to automatically follow a model in the Gazebo world as it moves. The behavior of the plugin is controlled using **dynamic_reconfigure**. The reconfigurable parameters for the plugin are shown in Table 2.

Param Name	Type	Description
enable	bool	Enable automatic camera control. If this is unchecked, the Gazebo camera behaves normally.
model_name	string	This is the name of the Gazebo model to follow with the camera. If the model name doesn't exist or isn't spawned yet, the Gazebo camera behaves normally.
view_dist	float	This is the following distance of the Gazebo camera from the target model. The camera orientation can be changed manually, but the distance from the target model will be fixed to this value.
first_person	bool	Enable first-person view. If this is checked, the Gazebo camera is locked to a position and orientation relative to the base frame of the target model instead of just following the model's position in Gazebo world frame.
x_offset	float	x offset of first-person view relative to the target model's base frame.
y_offset	float	y offset of first-person view relative to the target model's base frame.
z_offset	float	z offset of first-person view relative to the target model's base frame.
view_angle	float	Pitch angle of first-person view relative to the target model's base frame.

Table 2: Dynamic reconfigure parameters for the camera control Gazebo plugin.

2.4 Supported Gazebo Models

Six models representing the different vehicles supported by the Dataspeed drive-by-wire system simulator are included. These are shown in Figure 1.

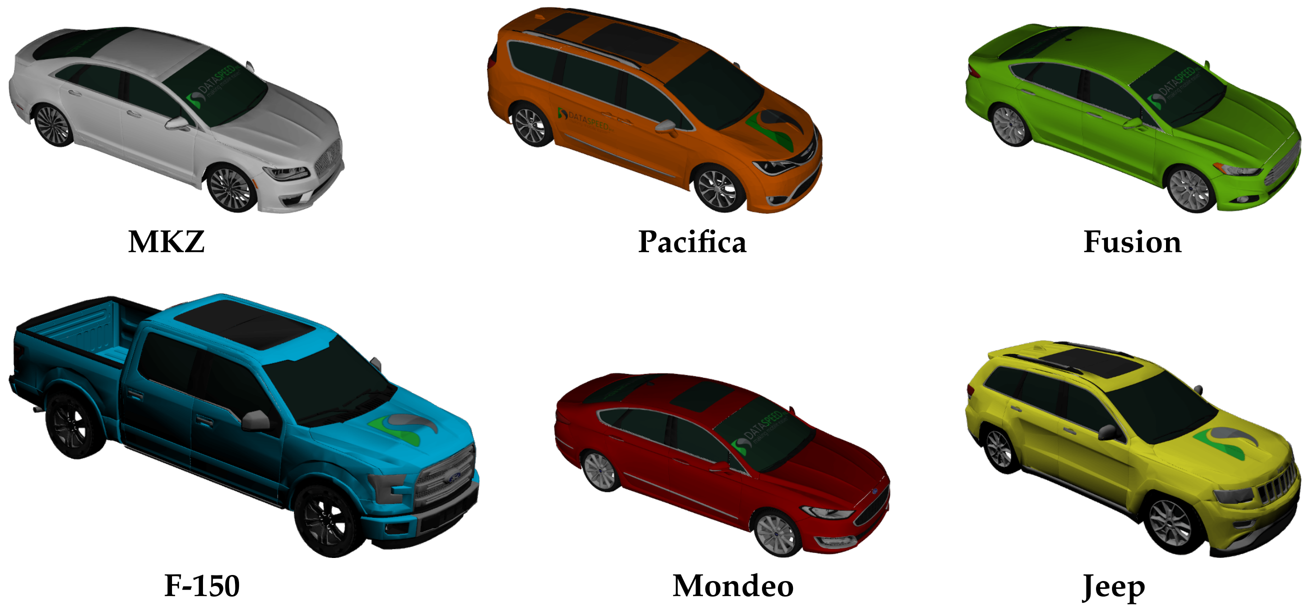

Figure 1: Models in the Dataspeed drive-by-wire system simulator.

2.5 ROS Topic Interface

The simulator emulates the CAN message interface to the real Dataspeed drive-by-wire system, and exposes the **can_bus_dbw/can_tx** topic to send CAN messages to the vehicle, and **can_bus_dbw/can_rx** to receive feedback data from the vehicle. Additionally, the user can publish the state of the physical turn signal lever on the **physical_turn_signal** topic to simulate a safety driver activating the physical turn signal. These topics and their corresponding message types are listed in Table 3.

The simulator only implements a subset of the complete Dataspeed CAN message specification. The supported command messages are listed in Table 4, and the supported report messages are listed in Table 5. See the Dataspeed drive-by-wire system datasheets for complete CAN message information.

The physical turn signal message is of type **std_msgs/Int32**, with 0 representing no active turn signal, 1 representing an active left turn signal, and 2 representing an active right turn signal.

Topic Name	Msg Type	Subscribes or Publishes
<topic namespace>/physical_turn_signal	std_msgs/Int32	Subscribes
<topic namespace>/can_bus_dbw/can_rx	can_msgs/Frame	Subscribes
<topic namespace>/can_bus_dbw/can_tx	can_msgs/Frame	Publishes

Table 3: ROS topics to interact with simulated Dataspeed drive-by-wire system.

Command Msg	CAN ID
Brake	0x060
Throttle	0x062
Steering	0x064
Gear	0x066
Misc (Turn signals only)	0x68
ULC Command	0x076
ULC Config	0x077

Table 4: Command CAN messages supported by the Dataspeed drive-by-wire system simulator.

Report Msg	CAN ID	Data Rate
Brake	0x061	50 Hz
Throttle	0x063	50 Hz
Steering	0x065	50 Hz
Gear	0x067	20 Hz
Misc	0x069	50 Hz
Wheel Speed	0x06A	100 Hz
Accel	0x06B	100 Hz
Gyro	0x06C	100 Hz
GPS1	0x06D	1 Hz
GPS2	0x06E	1 Hz
GPS3	0x06F	1 Hz
Brake Info	0x074	50 Hz
ULC Report	0x078	5 or 50 Hz

Table 5: Report CAN messages supported by the Dataspeed drive-by-wire system simulator.

2.6 Difference Between Models

Depending on which model is used, the Gazebo plugin emulates the appropriate Dataspeed drive-by-wire system firmware, meaning that:

- MKZ, Fusion, and Mondeo models all report themselves as FORD_CD4 Dataspeed drive-by-wire system platforms
- F-150 models report a FORD_P5 platform
- Pacifica models report a FCA_RU platform
- Jeep models report a FCA_WK2 platform

The Gazebo plugin emulates the different platforms by sending the same platform and version CAN messages as the real firmware does. Beyond this, the only difference between the models are some subtle wheel speed measurement deadband differences that are reflected in the production CAN bus information.

3 Configuration YAML Files

The parameters of the Dataspeed drive-by-wire system Gazebo simulation are set using a single YAML file. This section describes the options and formatting of the YAML file.

3.1 Available Parameters

The parameters shown in Table 6 can be specified in a YAML file to control the behavior of a simulated vehicle.

Param Name	Default	Description
x	0.0	x coordinate of the spawn location of the model.
y	0.0	y coordinate of the spawn location of the model.
z	0.0	z coordinate of the spawn location of the model.
yaw	0.0	Initial heading angle of the vehicle, relative to Gazebo world frame.
start_gear	'D'	Single character indicating the starting transmission gear. Must be one of 'P', 'R', 'N', 'D', or 'L'. Others will result in the default of 'D'.
color	'silver'	Color of the vehicle body. See Section 3.4 for details on the behavior of this parameter.
pub_tf	false	Publish a tf transform from the footprint frame of the vehicle to 'world' frame using the true pose of the vehicle in Gazebo world frame.
pub_odom	false	Publish a nav_msgs/Odometry message with the true pose of the vehicle in Gazebo world frame, and the vehicle's true velocity and yaw rate.
model	'mkz'	Model of vehicle to spawn. Must be one of 'mkz', 'fusion', 'mondeo', or 'pacifica'. Others will result in the default of 'mkz'.
production_ref_lat	45.0	Reference latitude for the simulated production GPS receiver.

production_ref_lon	-81.0	Reference longitude for the simulated production GPS receiver.
topic_ns	vehicle	Topic and node namespace for everything related to the model.
tf_prefix	”	TF frame prefix. Defaults to no prefix.
urdf_package	dataspeed_ dbw_gazebo	ROS package containing the simulated model's URDF file.
urdf_path	urdf/default_ model.urdf.xacro	Relative path from the urdf_package parameter to the actual URDF file.

Table 6: Dataspeed drive-by-wire system simulation configuration parameters.

3.2 Dictionary Format

The parameters listed in Table 6 must be set in an array of YAML dictionaries, where each dictionary corresponds to a single simulated vehicle. The name of the dictionary becomes the name of the spawned model. Consider the following YAML file:

```yaml
- vehicle:
    x: 5.0
    y: 20.0
    z: 0.0
    yaw: 0.0
    start_gear: D
    pub_tf: true
    pub_odom: true
```

This would simulate a single MKZ whose model name is **vehicle**. It would spawn at coordinates $(5.0,\ 20.0,\ 0.0)$ with a heading of 0.0, and the transmission would start in drive. The simulation would publish both a nav_msgs/Odometry message and a TF frame transform to indicate the vehicle's true pose and velocity in Gazebo world frame. The unspecified parameters would be the default values shown in Table 6.

3.3 Simulating Multiple Vehicles

To simulate multiple vehicles simultaneously, simply add more dictionaries to the array in the YAML file. Below is an example:

```
- vehicle1:
    x: 0.0
    y: -2.0
    color: red
    model: mkz
- vehicle2:
    x: 0.0
    y: 2.0
    color: green
    model: fusion
```

This would spawn two vehicles: one red MKZ with model name **vehicle1** spawned at $(0.0,\ -2.0,\ 0.0)$ and one green Fusion with model name **vehicle2** spawned at $(0.0,\ 2.0,\ 0.0)$. Both vehicles would have the default values of the parameters not set in the individual dictionaries.

3.4 Body Color

The **color** parameter is used to change the paint color of the simulated vehicle. It can be set as either a string or a dictionary with entries **r**, **g**, and **b**. Using a string selects between a number of preset colors, which are:

silver red green blue pink white black

Any other color strings will result in the default silver color. However, if the color parameter is set using a dictionary, any arbitrary color can be used. Below is an example YAML file that spawns one vehicle with the preset green color, and another with r=0.1, g=0.2, b=0.3:

```
- mkz1:
    color: green
- mkz2:
    color: {r: 0.1, g: 0.2, b: 0.3}
```

If the color dictionary is malformed, the default silver color is used instead.

3.5 Using Custom URDF Files

To use a different URDF model than the default model provided in the installed **dataspeed_dbw_gazebo** package, the user would set the **urdf_package** and **urdf_path** simulation configuration parameters to their particular URDF file. The **urdf_package** parameter is the name of a ROS package where the URDF file is stored, and the **urdf_path** parameter is the relative path within that ROS package to the particular file. For example, if the custom URDF file is at an absolute path of:

/home/username/catkin_ws/src/example_package/urdf/custom.urdf.xacro

then the simulation parameters should be set like this:

```
- vehicle:
    urdf_package: example_package
    urdf_path: urdf/custom.urdf.xacro
```

4 Example Simulation Package

A ROS package is provided that contains example simulation configurations to demonstrate how the Dataspeed drive-by-wire Gazebo simulator can be used to develop applications. This package is called **custom_dbw_sim_examples** and can be found in the public repository:
`https://bitbucket.org/dataspeedinc/dataspeed_dbw_simulation`

4.1 Demo Simulations

The **launch** folder of the **custom_dbw_sim_examples** package provides launch files to bring up three different demonstration simulations.

4.1.1 joystick_demo_sim.launch

The **joystick_demo_sim.launch** file simulates the same joystick demo program that can be run with the real Dataspeed drive-by-wire system. It is a standalone launch file that brings up all necessary nodes at once. A screenshot of the default joystick demo world is shown in Figure 2.

The launch file has two arguments, **model** and **dev**. The model argument defaults to `'mkz'` and allows the user to change which vehicle model is loaded when the simulation starts. Possible values are:

mkz   fusion   mondeo   f150   pacifica   jeep

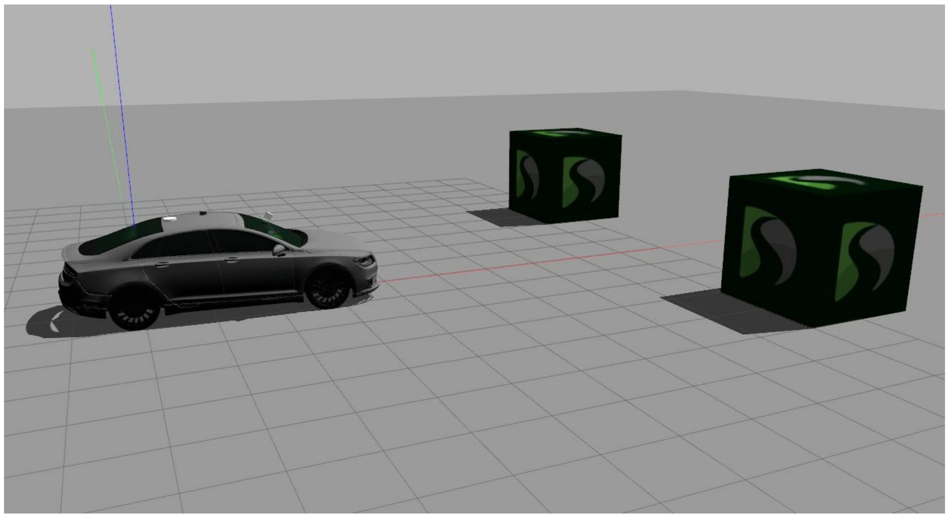
Figure 2: Joystick demo simulation world.

The dev argument defaults to `/dev/input/js0` and represents the physical joystick device connected to the computer running the simulation.

4.1.2 multi_car_sim.launch

The **multi_car_sim.launch** file demonstrates how multiple vehicles can be simulated at the same time. It is a standalone launch file that spawns both a Fusion and a Pacifica model and publishes separate speed and yaw rate commands to each one. A screenshot of the example multi-car simulation is shown in Figure 3.

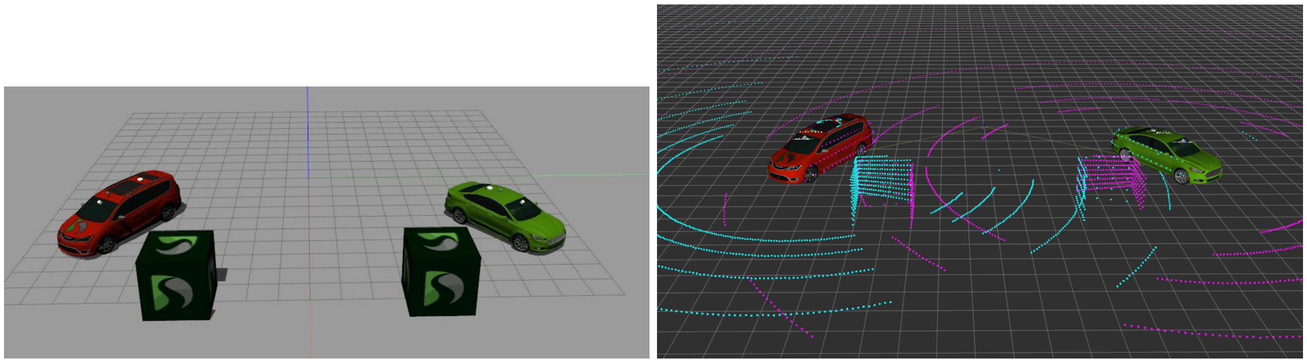
Figure 3: Gazebo and Rviz displays while running the demo multi-car simulation.

4.1.3 lane_keep_demo.launch

The **lane_keep_demo.launch** file demonstrates a more complete AV system than the other examples, and how simulated sensors can be mounted to the vehicle, and how the sensor data can be processed and used to control the vehicle. It is a standalone launch file that brings up the simulation, an image processing pipeline to extract lane lines, and a path following control algorithm. The image processing pipeline finds the center of the lane, and the path following controller generates Universal Lat/Lon Controller (ULC) commands to follow the lane. A screenshot of the lane keeping simulation is shown in Figure 4.

The ULC is a feature released in the Dataspeed drive-by-wire system firmware in December 2018, which adds a higher level of control over the actuator interface The ULC allows users to specify speed commands with acceleration limits instead of controlling the pedals directly, and it allows users to specify a desired yaw rate or turning radius instead of controlling the steering wheel angle directly. This lane keeping demo demonstrates how a user can send commands to the ULC using the **dataspeed_ulc_ros** node.

For more details about the ULC, the User's Guide for the feature is available on the Downloads page of its ROS interface repository:
`https://bitbucket.org/DataspeedInc/dataspeed_ulc_ros/downloads`

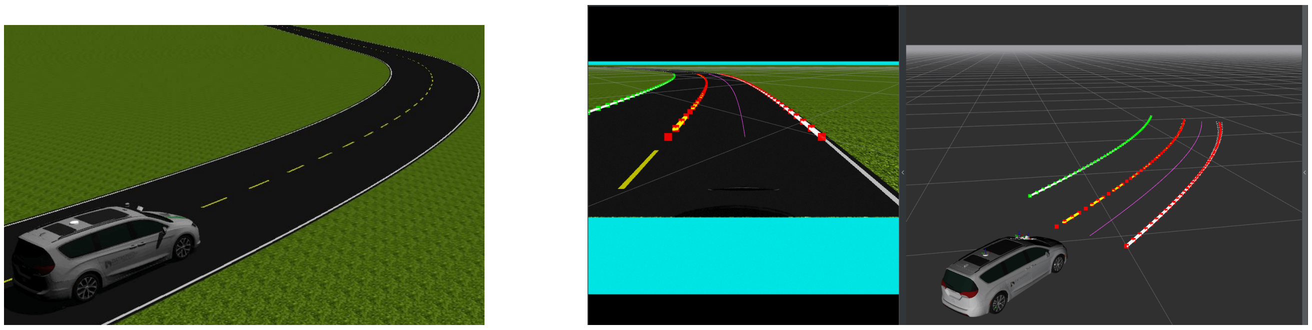

Figure 4: Gazebo and Rviz displays while running the demo lane keeping simulation.

The launch file has a single argument **model**. The model argument defaults to '`pacifica`' and allows the user to change which vehicle is loaded when the simulation starts. Possible values are:

mkz fusion mondeo f150 pacifica jeep

4.2 Configuration YAML Files

Sample configuration files for the demo simulations shown in Section 4.1 are found in the **yaml** folder of the **custom_dbw_sim_examples** package. Each of the files ending in **_joystick.yaml** are the model-specific configuration files for the joystick demo simulation. Each of the files ending in **_test_track.yaml** are the model-specific configuration files for the lane-keeping simulation. **multi_car.yaml** is for the multi-vehicle simulation.

4.3 URDF Models

Sample URDF files for each of the supported Gazebo models can be found in the **urdf** folder inside the **custom_dbw_sim_examples** package. Each type of vehicle has its own master URDF file, which is the one that is parsed upon loading the model into Gazebo:

- **f150_model.urdf.xacro**
- **fusion_model.urdf.xacro**
- **jeep_model.urdf.xacro**
- **mkz_model.urdf.xacro**
- **mondeo_model.urdf.xacro**
- **pacifica_model.urdf.xacro**

These are all identical except for small differences in where the sensors are mounted on the vehicle. The actual model that is spawned in Gazebo is determined by the **model** parameter in the configuration YAML file.

4.4 Sensors

A GPS and front-facing camera are included with the simulator as examples. They are defined by xacro macros that are found in **vehicle_sensors.urdf.xacro** in the **custom_dbw_sim_examples** package. When included, the example GPS sensor macro adds a white disk where it is mounted, and the example camera macro adds a white box. This is shown in Figure 5.

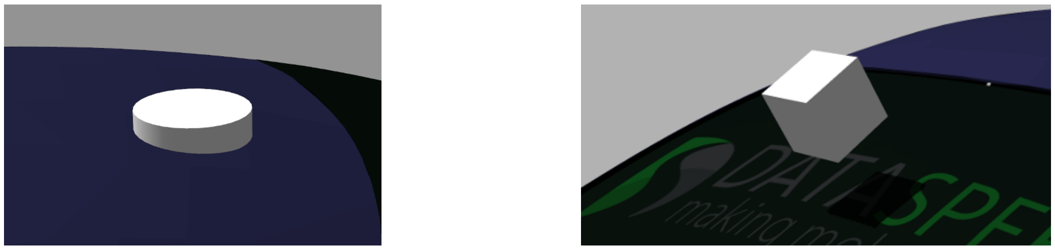

Figure 5: Visual models of the example GPS sensor (left), and the example camera (right).

4.4.1 GPS Receiver

The example GPS sensor built into the Dataspeed drive-by-wire system simulator provides perfectly accurate GPS position and heading measurements based on the vehicle model's position in the Gazebo world frame. The macro's parameters are described in Table 7. The GPS simulation plugin provides data on a collection of ROS topics that are advertised in the plugin's namespace. These topics are outlined in Table 8.

Param Name	Type	Units	Description
name	string	—	Name of the sensor. This defines the link name of the sensor, its corresponding TF frame name, and the namespace of the simulated data topics.
parent	string	—	Name of the link to which the sensor is attached.
x	float	meters	x offset of the sensor link, relative to **parent**.
y	float	meters	y offset of the sensor link, relative to **parent**.
z	float	meters	z offset of the sensor link, relative to **parent**.
rate	float	Hz	Rate at which to publish simulated GPS position and heading measurement messages.
ref_lat	float	degrees	Latitude corresponding to point $(0, 0, 0)$ in Gazebo world frame.
ref_lon	float	degrees	Longitude corresponding to point $(0, 0, 0)$ in Gazebo world frame.
ref_alt	float	meters	Altitude corresponding to point $(0, 0, 0)$ in Gazebo world frame.

Table 7: Gazebo GPS plugin parameters.

Topic Name	Msg Type	Description
fix	sensor_msgs/NavSatFix	Simulated position in latitude, longitude, altitude.
enhanced_fix	gps_common/GPSFix	More detailed version of **fix** message, including the heading w.r.t true North.
odom	geometry_msgs/Vector3Stamped	Simulated position in UTM coordinates. Zone and hemisphere boundary transitions are simulated.
vel	geometry_msgs/Vector3Stamped	Simulated velocity vector. x velocity component is relative to Due East, y velocity component is relative to due North.
heading	std_msgs/Float64	Simulated heading angle in NED frame. Angle is represented in degrees and wrapped between 0 and 360.

Table 8: Simulated GPS measurement ROS topics.

4.4.2 Camera

The example camera sensor built into the Dataspeed drive-by-wire system simulator is a xacro macro that runs a standard Gazebo camera simulation. Besides creating the simulated camera, the xacro macro sets up URDF links for the camera sensor and corresponding optical frame, and provides parameters that allow the camera to be easily attached to the main vehicle's URDF model. The macro parameters are listed in Table 9.

Param Name	Type	Units	Description
name	string	—	Name of the sensor. This defines the link name of the sensor, its corresponding TF frame name, and the namespace of the simulated data topics.
parent	string	—	Name of the link to which the sensor is attached.
x	float	meters	x offset of the sensor link, relative to **parent**.
y	float	meters	y offset of the sensor link, relative to **parent**.
z	float	meters	z offset of the sensor link, relative to **parent**.
roll	float	radians	Roll angle relative to parent.
pitch	float	radians	Pitch angle relative to parent.
yaw	float	radians	Yaw angle relative to parent.

Table 9: Gazebo camera plugin xacro macro parameters.