Lab Guide
Vehicle Control

Background

The objective of the vehicle control lab guide is to teach students about how the vehicle speed and steering controller architecture can be implemented. Prior to starting this lab guide please review the following concept reviews:

- Concept Review – Intro to Control
- Concept Review – PID Control
- Concept Review – Longitudinal Speed Control
- Concept Review – Geometric Lateral Control

Considerations for this lab guide:

Vehicle Control State is highly dependent on how the frame of reference of the moving body is defined. Concept Review – Geometric Control uses the center of the rear axel as the reference frame for formulating the steering control functions which are implemented in Quanser's solutions for the following skills activities.

This lab will ask you to complete 2 pipelines as shown below:

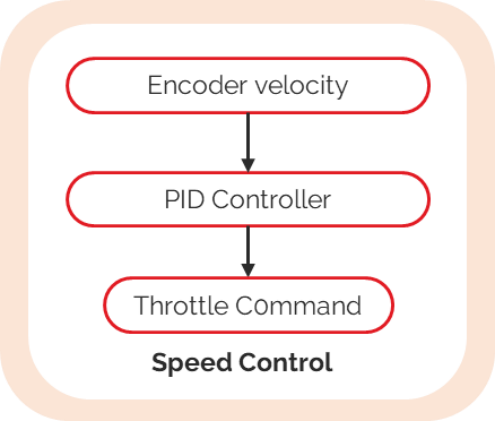

Figure 1: Speed Control

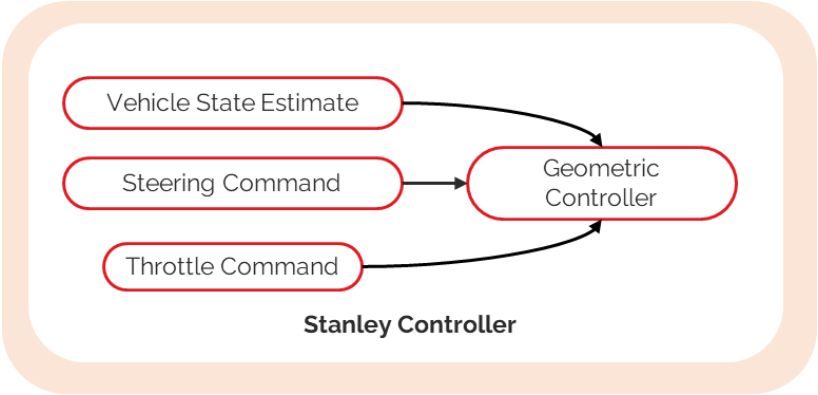

Figure 2: Geometric Lateral Control

Skills Activity - Setup

The physical setup for the lab will require two components:

- QCar stand
- SDCS road map

Review Setup Guide – Instructor for information on the size of the roadmap. The default structure of this map will rely on the large map configuration. Students will focus on 2 skills activities: speed and steering, respectively. To easily tune the gains utilized in the two control labs the speed control parameters and the steering control parameters have been laid out accordingly.

```
#================ Experiment Configuration ================
# ===== Timing Parameters
# - tf: experiment duration in seconds.
# - startDelay: delay to give filters time to settle in seconds.
# - controllerUpdateRate: control update rate in Hz. Shouldn't exceed 500
tf = 30
startDelay = 1
controllerUpdateRate = 100

# ===== Speed Controller Parameters
# - v_ref: desired velocity in m/s
# - K_p: proportional gain for speed controller
# - K_i: integral gain for speed controller
v_ref = 0.5
K_p = 0
K_i = 0

# ===== Steering Controller Parameters
# - enableSteeringControl: whether or not to enable steering control
# - K_stanley: K gain for stanley controller
# - nodeSequence: list of nodes from roadmap. Used for trajectory generation.
enableSteeringControl = False
K_stanley = 1
nodeSequence = [9, 14, 9]
```

When Skills Activity – Speed has been completed modify the **enableSteeringControl** from **False** to **True**. The steering control lab also takes into account the QCar's driving style, which defines on which side of the road the QCar is located. To understand how the node sequences work please review the following diagrams:

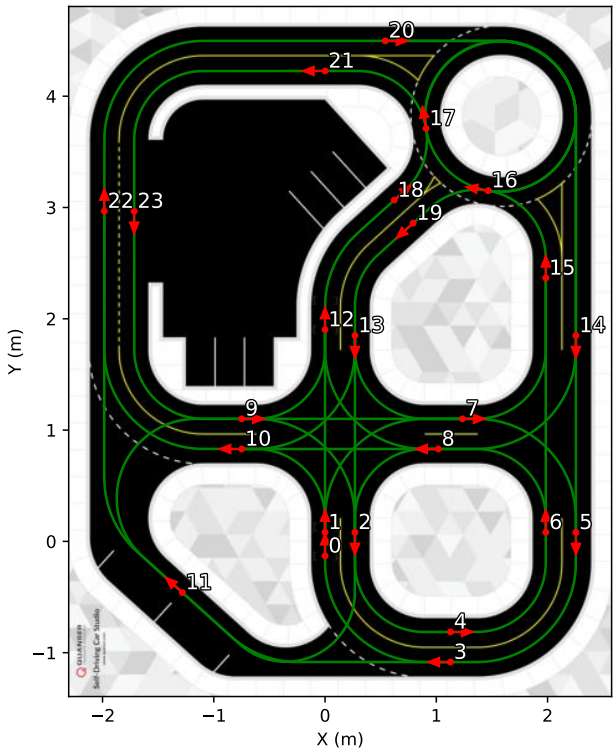

Figure 3: Driving style left- side of road.

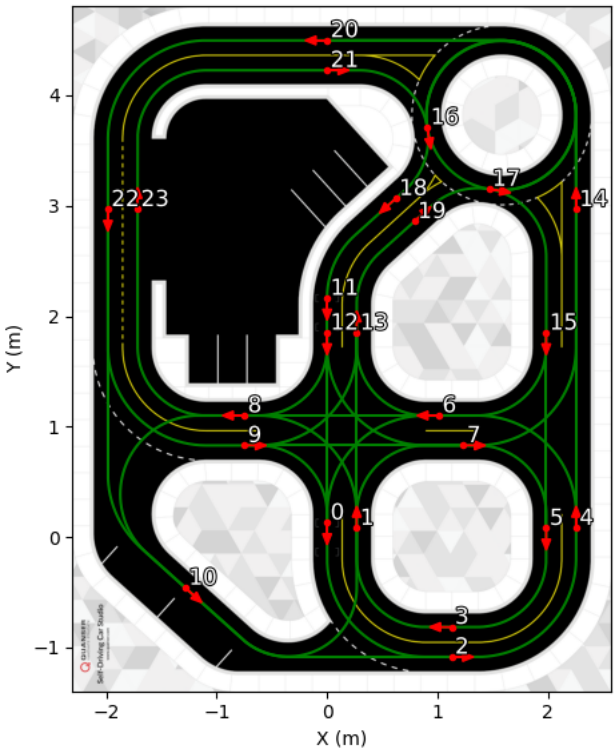

Figure 4: Driving style right-side of road.

Understanding the node sequence is important for the steering controller and defining which nodes the QCar will travel to. Variable **nodeSequence** is used to generate the waypoints the QCar will travel to. By default the QCar will travel in the sequence **[9, 14, 9]** to complete a full loop. Note: nodes **0** and **11** (**12** for left-hand drive) line up with the calibration points on the SDCS floor mats, giving an easy to identify starting location.

Skills Activity 1 – Speed Control

The objective of this section is to focus on the speed control architecture for an autonomous vehicle. What are some inputs/outputs to consider:

- QCar tachometer gives an indication of the vehicle speed over time (m/s)
- Motor throttle command defines the desired speed in m/s the QCar should travel

Using the concept reviews:

- Concept Review – Intro to Control
- Concept Review – PID Control
- Concept Review – Longitudinal Speed Control

Students develop an intuition on how a speed controller can be designed to match a desired speed.

Step 1 – Developing Longitudinal Speed Controller

Class **SpeedController()** has been initialized to consider the inputs:

- k_ref: Reference speed for QCar
- kp: Proportional Gain for speed control
- ki: Integral gain for speed control

Complete **SECTION A** of **SpeedController()** to utilize a configuration of a PID controller. For now, gains **P** and **I** have been initialized although it's encouraged to also include a **D** term to improve the vehicle velocity tracking performance. The **update** method takes $\boldsymbol{v_{ref}}$ which is the vehicle desired velocity, $\boldsymbol{dt}$ or the time step for the control loop and $\boldsymbol{v}$ which is the current linear velocity measurement of the QCar. For safety it's recommended to clip the maximum control command to 0.5 m/s. Change the **return** to be the commanded throttle calculated by the longitudinal speed controller.

```
    # ==============  SECTION A -  Speed Control  ====================
    def update(self, v, v_ref, dt):
        '''
        return np.zeros(1)
```

Physical setup -

Place the QCar on top of the QCar stand to prevent it from moving in the physical space. Review the User Manual – Connectivity for establishing a network connection to the QCar. Open a terminal window in the folder where the vehicle_control.py file is located:

If using a QCar 1, sudo authority is needed, run the following command:

```
sudo PYTHONPATH=$PYTHONPATH python3 vehicle_control.py
```

If using a QCar 2, run the following command:

```
python vehicle_control.py
```

Virtual setup -

Inside Quanser Interactive Labs navigate to the QCar Plane workspace prior to starting the skills activity. In a terminal session, navigate to the folder containing vehicle control skills activity. Note that currently a QCar 1 will be created, the spawn model for QCar 2 will be release in a future update. Use the following command to run the skills activity:

```
python vehicle_control.py
```

Results for Speed Control Lab:

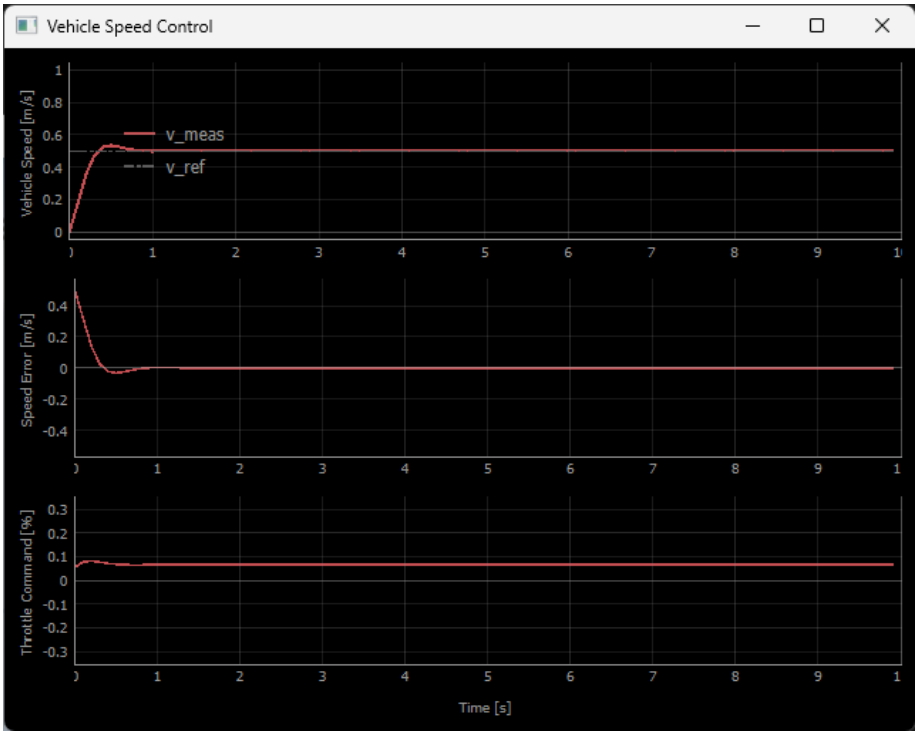

Figure 4: Results of Skills Activity 1 on virtual QCar.

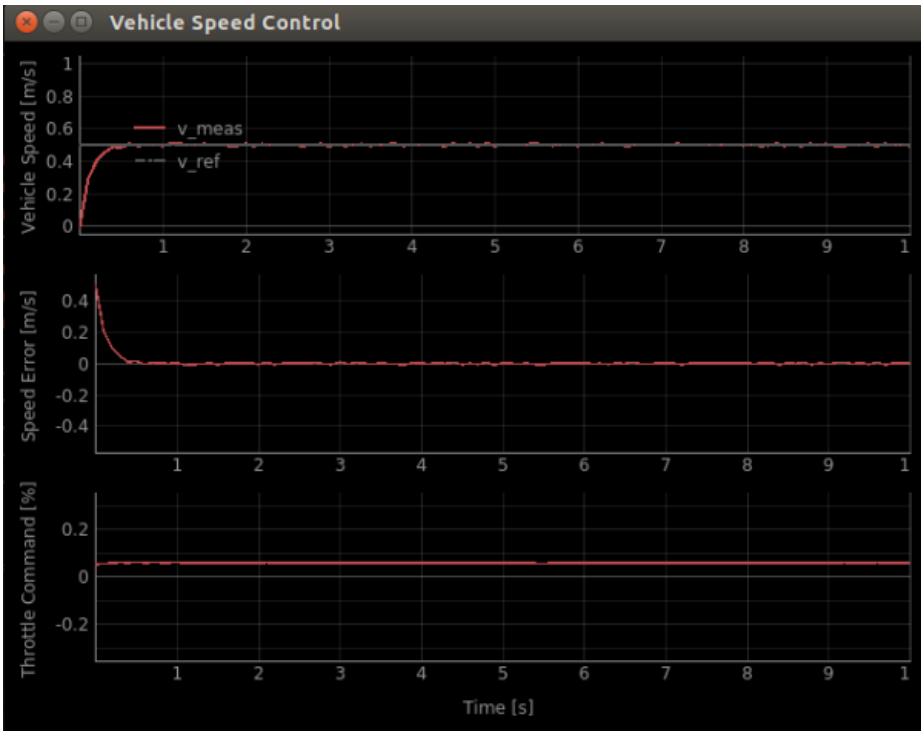

Figure 5: Results of Skills Activity 1 on Physical QCar.

Considerations

On the physical QCar make a note of the trajectory estimated using dead reckoning. How successful was the estimator at tracking the QCar as it performed a circular trajectory?

Skills Activity 2 – Geometric Steering Controller

The objective of this section is to develop the steering controller utilized by an autonomous vehicle. In Concept Review – Geometric Lateral Control two methods for steer control were presented. The goal for this section is to implement a Stanley steering controller. Students are encouraged to implement both a Pure Pursuit and a Stanley controller to learn about the performance differences between the two.

Note: if the skill activity will be performed on the virtual QCar, please review the Setup Guide for students on how to configure the directory structure to use **Quanser Interactive Labs**.

In this lab, you will carry out the following steps,

1. Implement a steering controller for the QCar

Step 1 – Implementation of Steering Controller

In the setup section the variable **enableSteeringControl** was originally set to **False**, change this value to **True** to enable the steering controller. Complete **SECTION B** of **SteeringController()** to implement a Stanley steering controller. Variables to consider:

- **wp_1:** Current waypoint the QCar is in.
- **wp_2:** Next waypoint for the QCar to track.
- **p:** Current position of QCar
- **th:** Current heading estimate of QCar
- **speed:** Current speed measurement for QCar
- **self.k:** Variable used to pass through the value of **K_stanley** defined in the setup section of this lab.

For safety it's recommended to clip the maximum control command to 0.3 rad. Change the **return** to be the commanded steering value calculated by the geometric steering controller.

```
    # ==============  SECTION B -  Steering Control  ====================
    def update(self, p, th, speed):
        wp_1 = self.wp[:, np.mod(self.wpi, self.N-1)]
        wp_2 = self.wp[:, np.mod(self.wpi+1, self.N-1)]

        '''
        return 0
```

Physical setup -

Based on Figure 4, place the QCar on the starting node configured during the setup section of this lab. For these results the node sequences were **[9, 14, 9]**. Review the User Manual – Connectivity for establishing a network connection to the QCar. Open a terminal window in the folder where the vehicle_control.py file is located:

If using a QCar 1, sudo authority is needed, run the following command:

```
sudo PYTHONPATH=$PYTHONPATH python3 vehicle_control.py
```

If using a QCar 2, run the following command:

```
python vehicle_control.py
```

A prompt will show up asking to recalibrate the lidar data. Recalibrate the lidar if:

- This is the first time running the LiDAR
- There have been changes in the physical environment such as walls being moved or obstacles in the workspace.

Note: The QCar must be calibrated from node 0 or 11 facing the direction of the traffic flow. To do this, run vehicle_control.py and calibrate the QCar at node 0 or 11. Once calibration is complete, cancel the vehicle control by pressing 'ctrl+c'. Run vehicle control a second time with the QCar at the node you want to start from, but **don't** calibrate again.

Virtual setup -

Inside Quanser Interactive Labs navigate to the QCar Cityscape or QCar Cityscape Lite workspace prior to starting the skills activity. In a terminal session, navigate to the folder containing vehicle control skills activity. Use the following command to run the skills activity

```
python vehicle_control.py
```

Results for Steering Control Lab:

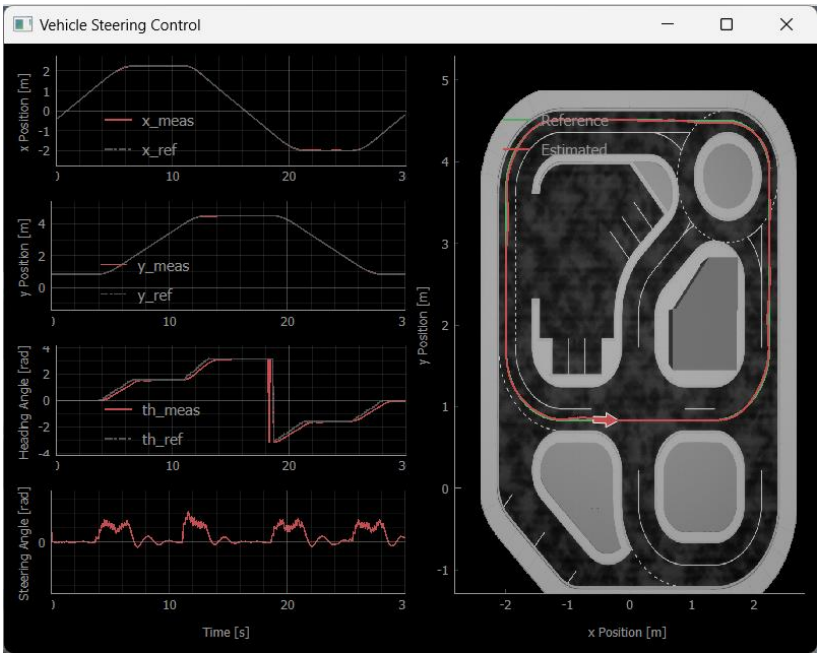

Figure 6: Results of Skills Activity 2 on virtual QCar.

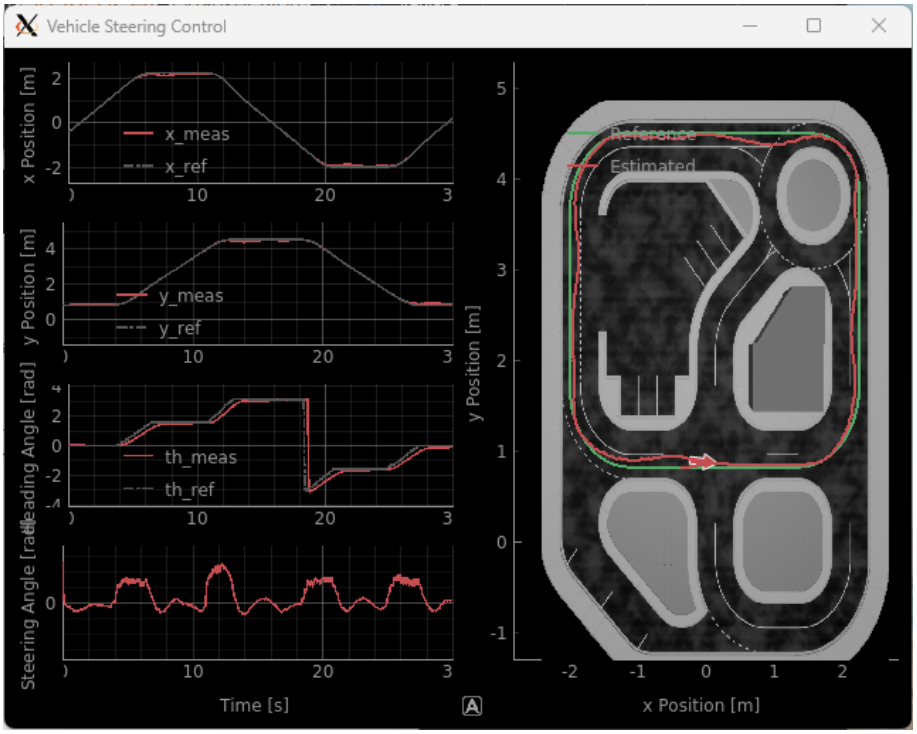

Figure 7: Results of Skills Activity 2 on physical QCar.

Considerations

How does changing the Stanley controller gain affect the performance of the QCar? If the Pure Pursuit algorithm was implemented, how did its' performance compare to a Stanley implementation?