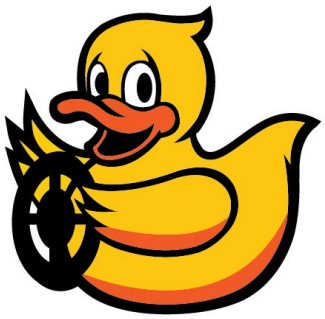
Setup Step 2.2.1
Wheels Calibration



Introduction

You might have noticed that your vehicle doesn't really go in a straight line when you command it to. Also, the vehicle might not go at the velocity you are commanding it to drive at.

This is due to the fact a slight difference between the motors and the wheels can cause the left wheel and right wheel to travel slightly different distance even though they have made the same rotation. Also, the system has no encoders, so we are commanding open loop voltages without ensuring the desired wheel velocity is met.

We can counter this behavior by calibrating the "gain" and "trim" on the commands that are sent to the wheels. This tutorial will walk you through the calibration process and also introduce the use of parameter server in ROS.

The inverse_kinematics_node under dagu_car pkg is in charge of translating a desired velocity and angular velocity command, also called Twist2D, to motor voltages. It also adjusting the wheel voltage commands by a gain and trim value. The relationship between the velocities and the output voltages are defined as:

> right_wheel_voltage = (gain + trim) * (linearVelocity + angularVelocity * 0.5 * baseline)
> left_wheel_voltage = (gain - trim) * (linearVelocity - angularVelocity * 0.5 * baseline)

The baseline is the distance between the two wheels. Note that if the gain = 1.0 and trim = 0.0, the wheel's voltages are exactly the same as the linear velocity + or - angular velocity times half the baseline length.

With gain > 1.0, the vehicle goes faster given the same velocity command, and for gain <1.0 it would go slower.

With trim > 0, the right wheel will turn slightly more than the left wheel given the same velocity command; with trim < 0, the left wheel will turn slightly more the right wheel.

Set the trim and gain

Make sure that your vehicle is on and connected to the wifi.

On your Duckiebot, launch the joystick

```
duckiebot: $ roslaunch duckietown_demos joystick.launch veh:=${VEHICLE_NAME}
```

Changing the trim to 0.01:

```
duckiebot: $ rosservice call /${VEHICLE_NAME}/inverse_kinematics_node/set_trim -- 0.01
```

Or Changing the trim in a negative way, e.g. to -0.01:

```
duckiebot: $ rosservice call /${VEHICLE_NAME}/inverse_kinematics_node/set_trim -- -0.01
```

Keep setting the trim until the duckiebot goes straight

Then start setting the gain:

```
duckiebot: $ rosservice call /${VEHICLE_NAME}/inverse_kinematics_node/set_gain -- 1.1
```

When you are all done, save the parameters by running:

```
duckiebot: $ rosservice call /${VEHICLE_NAME}/inverse_kinematics_node/save_calibration
```

If you do this the first time, you will see how it creates a new ${VEHICLE_NAME}.yaml file for your duckiebot in the folder:

duckietown/config/baseline/calibration/kinematics

which you can add and commit to the git repo.

AFTER THIS OLD - DO NOT USE####

You will see a warning message including the following line:

```
error loading <rosparam> tag:
file does not exist
[[...]/duckietown/config/baseline/calibration/wheels_trim/<VEHICLE_NAME>.yaml]
```

This is because when trim is enabled, joystick.launch will load the trim value from the file

~/duckietown/catkin_ws/src/duckietown/config/baseline/calibration/wheels_trim/${VEHICLE_NAME}.yaml.

So first we need to create such a file by first copying the default trim file under the same folder.

```
laptop: $ roscd duckietown/config/baseline/calibration/wheels_trim
laptop: $ cp default.yaml ${VEHICLE_NAME}.yaml
```

Now open <robot>.yaml. You should see:

```
trim: 0.0
```

With the file created you can now launch joystick with trim enabled

```
laptop: $ roslaunch duckietown joystick.launch veh:=${VEHICLE_NAME} trim:=true
```

Command the vehicle to go forward by pushing the left joystick forward. Observe the trajectory of the vehicle and take a note at which side does it turn to.

Using rosparam to file the trim value for you vehicle

Take a look of the current value of the trim using `rosparam get` in another terminal.

```
laptop: $ rosservice get /${VEHICLE_NAME}/wheels_trimmer_node/trim
```

The default value should be 0.0 as indicated by the veh_name.yaml file we just created.

You can set the trim value by rosparam set. Let's set it to an extreme value (0.5) to see how it affect the behavior of the vehicle:

```
laptop: $ rosparam set /${VEHICLE_NAME}/wheels_trimmer_node/trim 0.5
```

Command the vehicle to go forward using the left joystick, you should see the vehicle makes a rather sharp turn to the left, implying that the right wheel is traveling further than the left wheel.

Repeat the above mentioned process using rosparam set and find the trim value that will make your vehicle travel in a straight line. Remember this value.

Usually this value should be between [-0.05, 0.05], if the magnitude of the value is significantly outside of this set of bounds, or if your vehicle behaves differently every time even given the same trim value, you should inspect the wheels and motors of the your vehicle and make sure that the wires and the batteries are not touching (or anywhere near) the wheels.

Note that this trim value will not survive a relaunch of the joystick.launch. You can confirm this by killing the joystick.launch (Ctrl + C in the roslaunch terminal), and then relaunch (you bring back the last command by pressing up-arrow in the same terminal). Check the value of the trim parameter by

```
laptop: $ rosparam get /${VEHICLE_NAME}/wheels_trimmer_node/trim
```

The value should be 0.0 again.

To make the trim value persists between launches you need to write it to the `${VEHICLE_NAME}.yaml` file you just created. Edit the file:

```
laptop: $ roscd duckietown/config/baseline/calibration/wheels_trim
laptop: $ nano ${VEHICLE_NAME}.yaml
```

and change the value from 0.0 to the value that will make your vehicle drive in a straight line. Save the file.

Remember to commit the `${VEHICLE_NAME}.yaml` file when you're done.