PROGRAMOBOT

The Robotics club, IIT Palakkad

Objective

Participants should come-up with a control algorithm for making the bike self-balancing and capable of navigating to a given coordinate all by itself, without colliding anywhere. The bike is described in the section 'Robot Description' below. It will spawn inside a 'gas station', that will be its initial position. A destination coordinate is provided (see the 'Task' section) where the bike should stop within the mentioned threshold.

Installations

Link for packages

The packages are based on ROS Melodic (Gazebo 9)

Simply extract the given packages (inter_iit_sbb_description and sbb_dependencies) to your catkin workspace.

Now build your workspace after navigating to the src folder of the workspace by typing catkin build command. (*catkin make can also be done*)

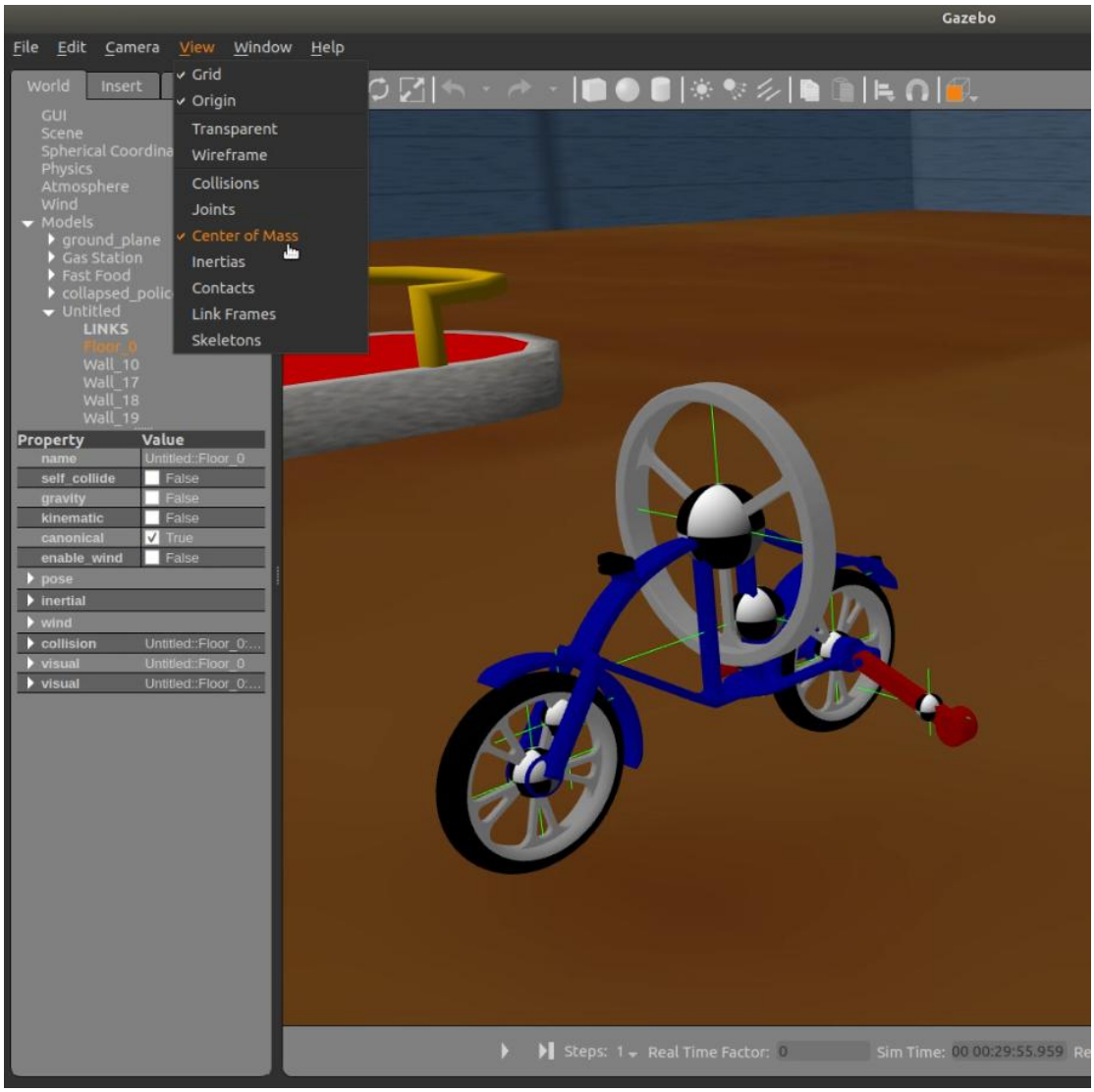
Go back one directory and execute - source devel/setup.bash for further convinience.

You are all-set to launch the simulation. There are two launch files in the launch directory - empty_world.launch and world.launch

First one launches the bike in an empty gazebo environment, this can be used to analyze the bike and test the control algorithm for balancing and some other intial tasks, as the rtf (real time factor) and fps (films per second) will be better in an empty world.

The task needs to be completed in the second launch file.

Make use of the 'view' feature in Gazebo to analyze the bike



Task

The bike will spwan inside a gas station, from there it has to autonomously navigate and reach the global coordinate - [latitude: 10.0001415852, longitude: 10.0003820894] , within a threshold window of 0.8 meters in x-axis and 0.5 meters in y-axis. (x,y axes are w.r.t to the image (top view)

shown below). Basically the bike should be parked inside the box like structure, without touching and walls.

The destination point is shown below

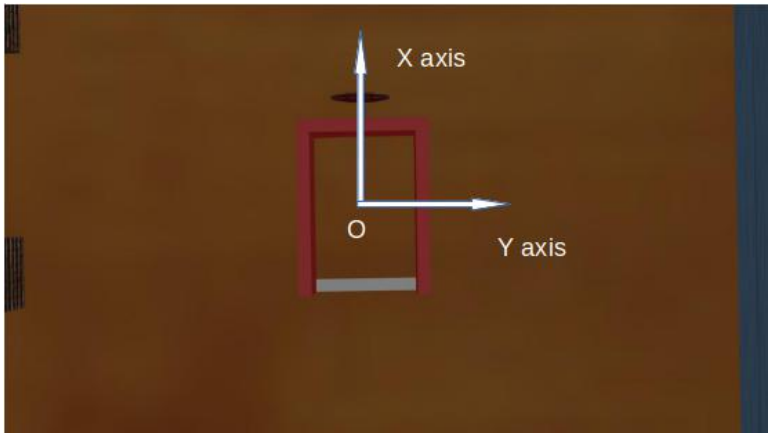
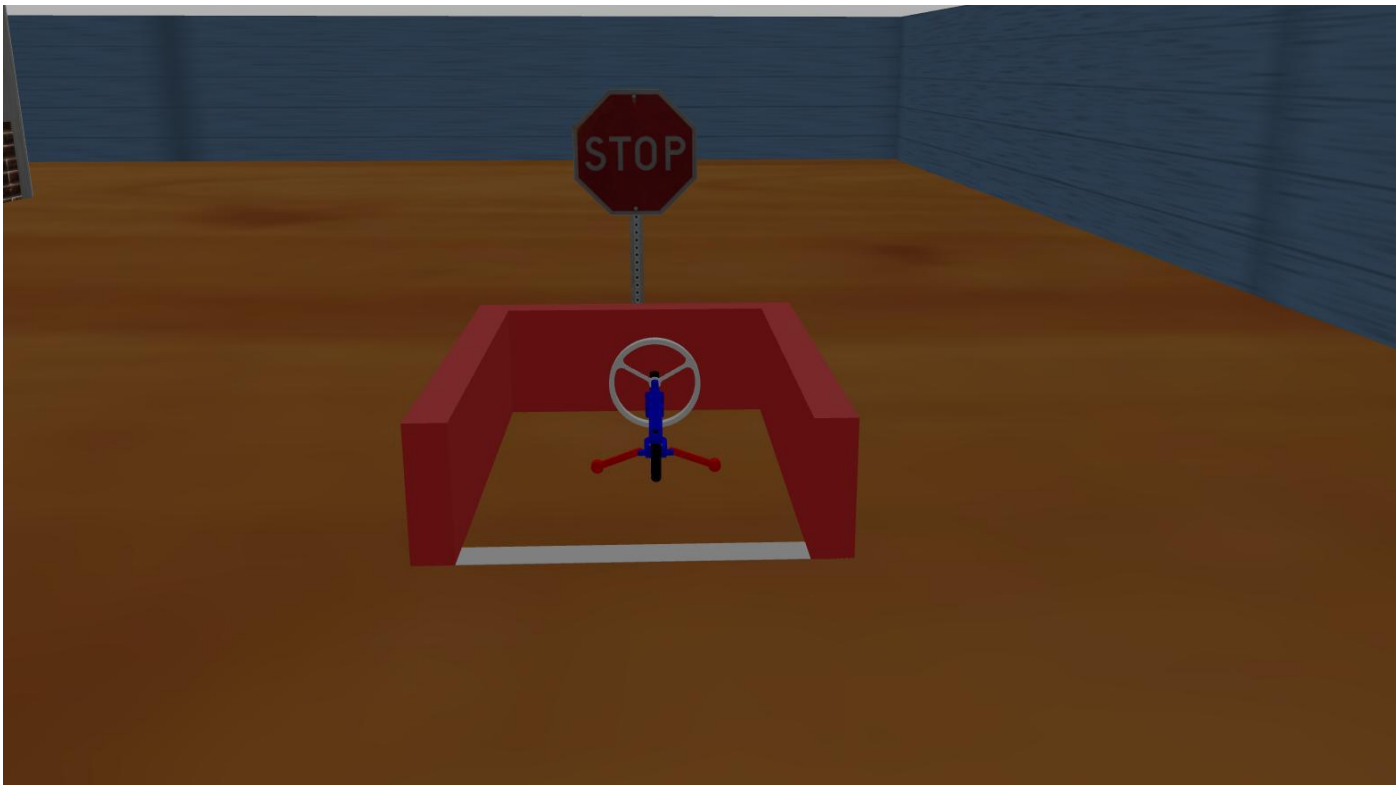


Fig : Top view

(The destination coordinates corresponds to point O)



Robot description

Traction - Rear wheel is responsible for traction.

Related topic - /drive_wheel/command

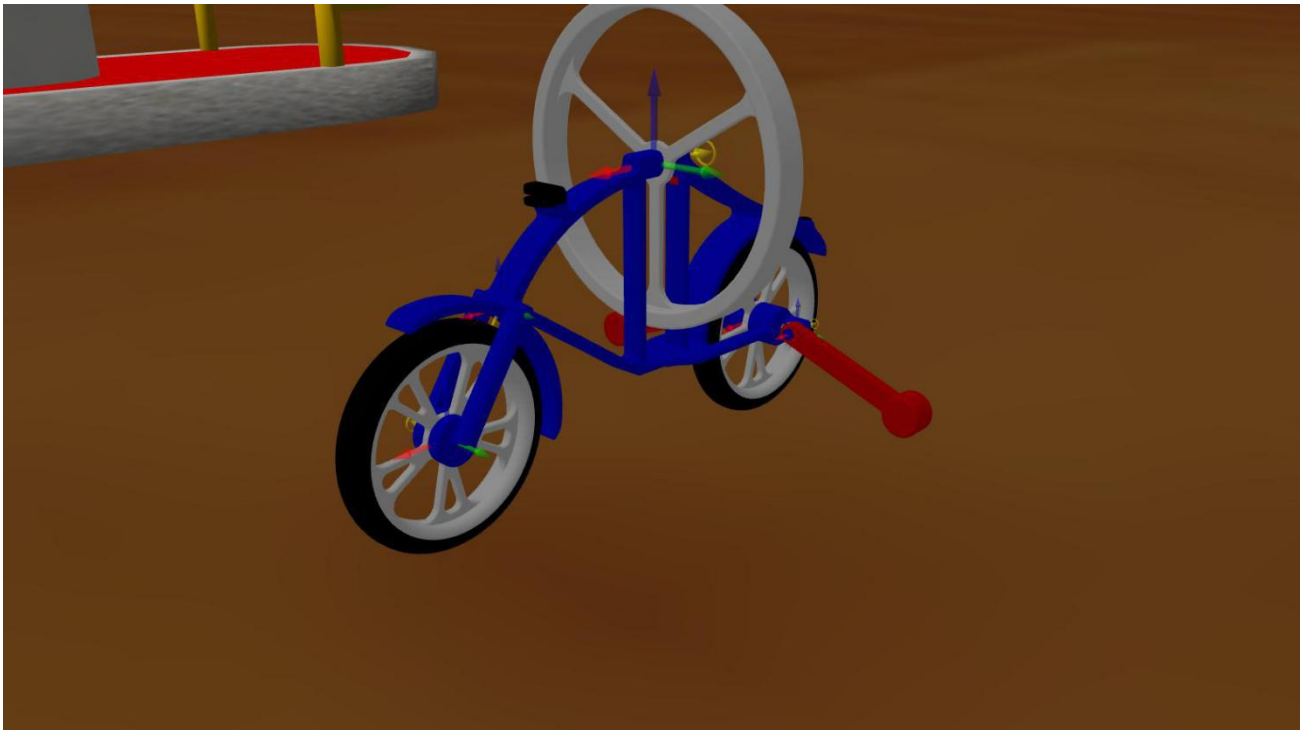
Direction (turning) - Front wheel is attached to a handle, capable of rotating w.r.t the main body for making cuts and turns.
Related topic - /handle/command

Balancing - A flywheel is attached to the main body to provide reaction torque to the bike for balancing. Also two stands (in red colour) is given to keep the bike upright when flywheel is not working. They can also rotate, hence can be used for balancing even when the flywheel is active.

Related Topics - Flywheel - /flywheel/command, Stands - /stand1/command and /stand2/command

Use the 'joints' option in the 'view' tool in Gazebo (mentioned above) to see the axes of all the revolute joints.

Rendered image of the bike



View in Gazebo

Actuators

Joint Motors -

All the 'revolute' joints in the bike, except for the front wheel, are equipped with joint motors and encoders. The joint motors are like ideal motors, hence there is no technical speed or torque limit. However you should limit the input speed to a certain value according to the requirement of your algorithm and mention it in comments in the top of the code.

Let's have a look at the topics related to the joint motors (drive_wheel is chosen only for explanation purpose, all other motors follow the same) -

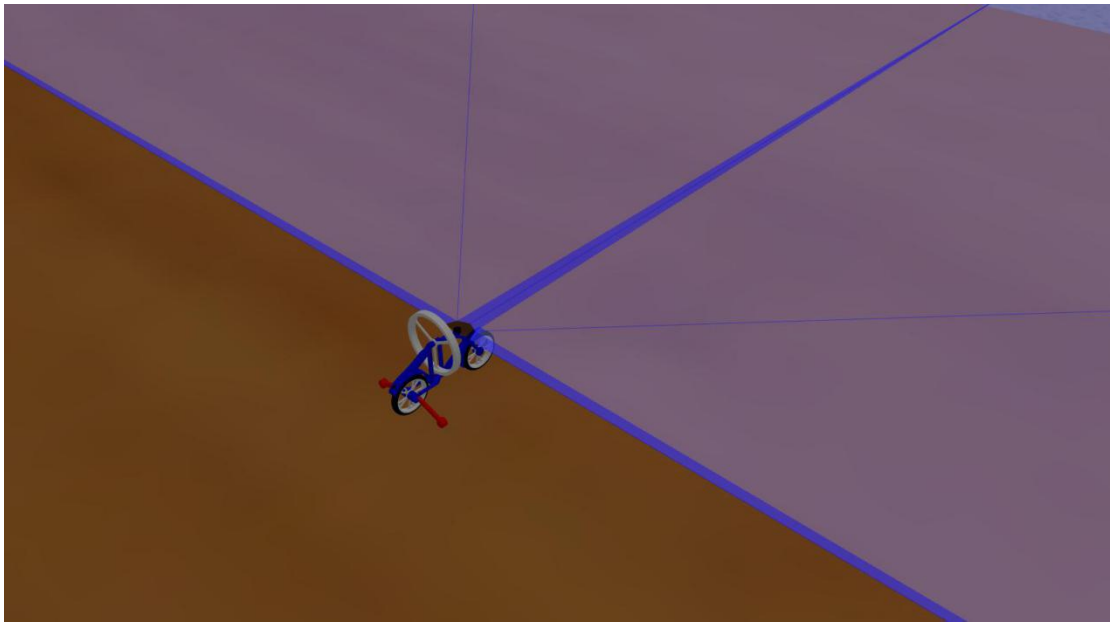
1. /drive_wheel/command - in this topic you can publish the required speed as a floating point number. Toggling the sign of input will change the direction of rotation.

2. /drive_wheel/joint_state - this topic contains the current state of the motor - position, velocity and effort. Subscribe the topic for making a closed loop control.

3. /drive_wheel/velocity - this is a renundant topic as the velocity is already available in the previous topic.

Sensors

1. LiDAR



The sensor has 5 sample points distributed over 180 degrees facing front of the bike. All the five readings are published as an array to the topic /sbb/distance_sensor/front.

You may need to figure out the exact direction correspinding to each sample point by physically placing obstacles (available in Gazebo GUI) and looking for changes in the values.

**sbb used in the topic names stands for self balancing bike.*

2. IMU

```
header:
  seq: 0
  stamp:
    secs: 1796
    nsecs: 569000000
  frame_id: "imu_link"
orientation:
  x: 0.00065065031743
  y: 5.0170704928e-05
  z: -2.88262218784e-05
  w: 0.999999786653
orientation_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
angular_velocity:
  x: 0.00408724370585
  y: -0.0203300261935
  z: 0.000665872839084
angular_velocity_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
linear_acceleration:
  x: 0.0623789158684
  y: 0.06062105368
  z: 9.80570474993
linear_acceleration_covariance: [0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]
---
```

The sensor will output following values

i. Orientation (in quarternions)
ii. Angular velocity
iii. Linear acceleration

Published to the topic - /sbb/imu

The sensor has some associated noise. Also checkout 'tf' library in Python for converting quarternions to euler angles.

3. GPS

The two topics of interest related to gps are /sbb/gps and /sbb/gps/vel rest can be looked upon as per requirement

/sbb/gps contains the global coordinates (latitude, longitude, altitude) of the robot. The altitude alone is in meters, rest two needs to be carefully converted/used.

/sbb/gps/vel contains the linear velocity in the three axes, again the units should be taken care of.

Apart from this all the joint motors (appearing in the topic list) have encoders that can be used for state estimation as well.

Making the submission

Required files -
i. Complete, unedited screen recording of the task (performed in world.launch)
ii. All the code (and related) files with proper commenting including explanation of functions and variables used.
iii. Video of explanation of the algorithm.

All these files should be zipped in a single folder and mailed to trc@iitpkd.ac.in
Subject of the mail should be - Submission for Programobot by (team name).
The mail body should contain mail id of all the team mates.

Rules

1. The descriptor file provided in the packages should not be modified in any sense. In case if the there is any issue, it should be informed to us, we will rectify that.

2. The default topics of gazebo should not be published or subscribed.

3. The path of the bike shouldn't be hard-coded.

4. The bike should start from the swpawned position itself, it cannot be translated or rotated using Gazebo GUI.

5. While recording for submission, all the tabs of terminal should be visible (hence better to use terminator and use the 'split horizontally/vertically' feature).

6. Any kind of plaglarism is stricly discouraged.

7. Organisers hold the rights to change/modify any aspect of the competition as per the requirement.

Judging and Scoring

Topics metioned in the previous document for event details (completion (simulation) time for the task taken by the bike, robustness of the algorithm for balancing and navigation) will be considered for judging, but unlike mentioned earlier, weightage of any sector is not pre-decided.

Join the discord server

https://discord.gg/H6ecNfdS (kindly put your nickname as - Your_name|Team_name)