SC-635 Advanced Topics in Mobile Robotics

Experiment Module : Control of Differential Drive in ROS

January 24, 2020

Systems and Control Engineering

Indian Institute of Technology Bombay

Overview

1. Gazebo : getting started

2. Turtlebot Operation

Recapitulation

- Topics
- Nodes
- Messages
- Launch File
- Passing arguments to nodes from launch file

Gazebo simulation environment

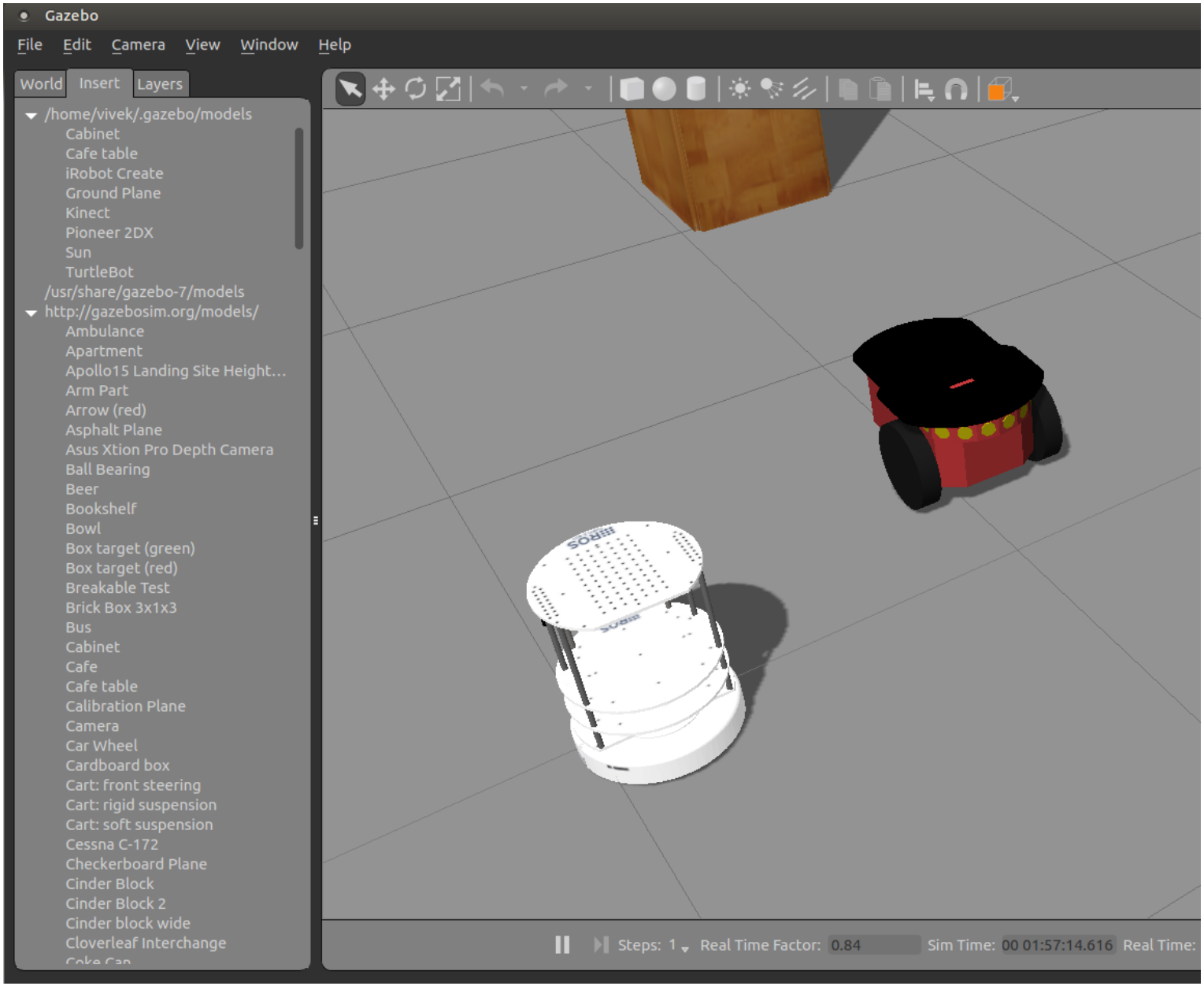
Features:

- Separate open source project that aims at
  making robot simulation easy
- 3D simulation
- Rigid body dynamics
- Open Dynamics, Bullet, Dart Engine
- Well integrated with ROS

find more information at

http://gazebosim.org/

Snapshot


Gazebo
File Edit Camera View Window Help
World Insert Layers
/home/vivek/.gazebo/models
Cabinet
Cafe table
iRobot Create
Ground Plane
Kinect
Pioneer 2DX
Sun
TurtleBot
/usr/share/gazebo-7/models
http://gazebosim.org/models/
Ambulance
Apartment
Apollo15 Landing Site Height...
Arm Part
Arrow (red)
Asphalt Plane
Asus Xtion Pro Depth Camera
Ball Bearing
Beer
Bookshelf
Bowl
Box target (green)
Box target (red)
Breakable Test
Brick Box 3x1x3
Bus
Cabinet
Cafe
Cafe table
Calibration Plane
Camera
Car Wheel
Cardboard box
Cart: front steering
Cart: rigid suspension
Cart: soft suspension
Cessna C-172
Checkerboard Plane
Cinder Block
Cinder Block 2
Cinder block wide
Cloverleaf Interchange
Steps: 1 Real Time Factor: 0.84 Sim Time: 00 01:57:14.616 Real Time:

Running Gazebo

- Need the package *gazebo_ros*
- Installed as a part of ros-kinetic-desktop-full
- Execute following command to bring up the simulator gui

```
$ rosrun gazebo_ros gazebo - -verbose
```

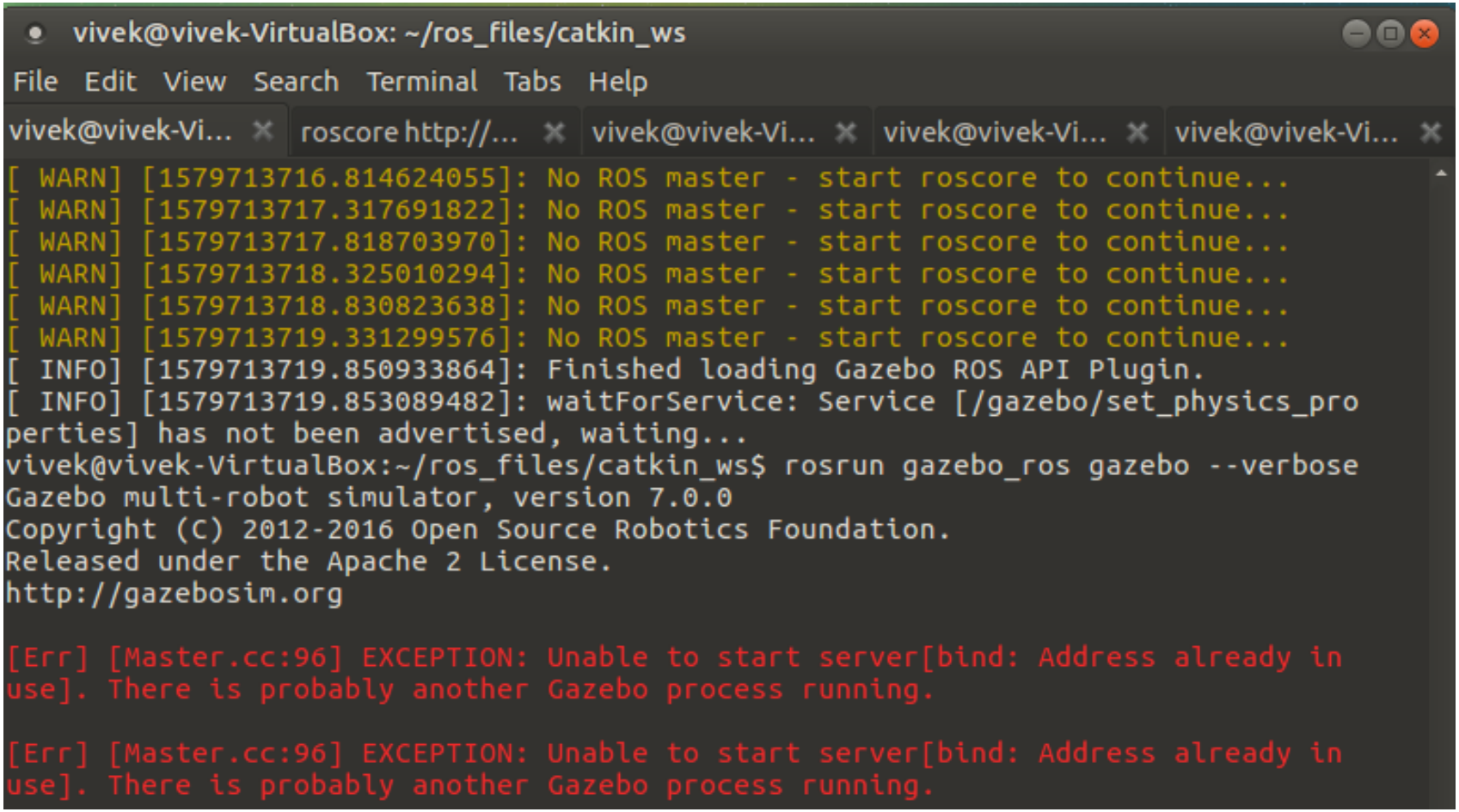
- ...

Running Gazebo

- Need the package *gazebo_ros*
- Installed as a part of ros-kinetic-desktop-full
- Execute following command to bring up the simulator gui

  $ rosrun gazebo_ros gazebo - -verbose
- ...
- Mostly, the above command won't work the first few times

Snapshot

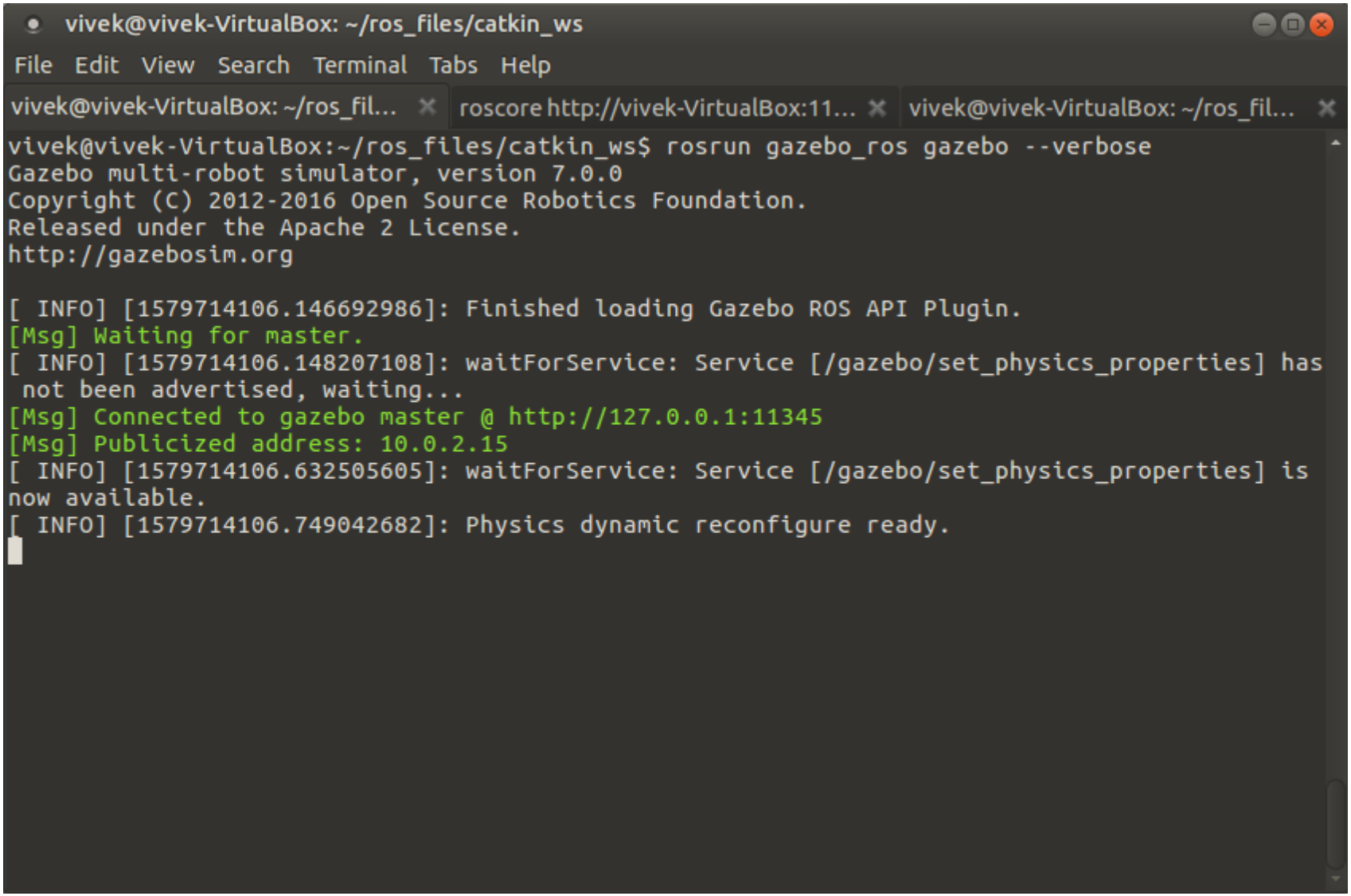

vivek@vivek-VirtualBox: ~/ros_files/catkin_ws
File Edit View Search Terminal Tabs Help
vivek@vivek-Vi...
roscore http://...
vivek@vivek-Vi...
vivek@vivek-Vi...
vivek@vivek-Vi...
[WARN] [1579713716.814624055]: No ROS master - start roscore to continue...
[WARN] [1579713717.317691822]: No ROS master - start roscore to continue...
[WARN] [1579713717.818703970]: No ROS master - start roscore to continue...
[WARN] [1579713718.325010294]: No ROS master - start roscore to continue...
[WARN] [1579713718.830823638]: No ROS master - start roscore to continue...
[WARN] [1579713719.331299576]: No ROS master - start roscore to continue...
[INFO] [1579713719.850933864]: Finished loading Gazebo ROS API Plugin.
[INFO] [1579713719.853089482]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
vivek@vivek-VirtualBox:~/ros_files/catkin_ws$ rosrun gazebo_ros gazebo --verbose
Gazebo multi-robot simulator, version 7.0.0
Copyright (C) 2012-2016 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazebosim.org
[Err] [Master.cc:96] EXCEPTION: Unable to start server[bind: Address already in use]. There is probably another Gazebo process running.
[Err] [Master.cc:96] EXCEPTION: Unable to start server[bind: Address already in use]. There is probably another Gazebo process running.

Running Gazebo

- Need the package *gazebo_ros*
- Installed as a part of the full desktop installation
- Execute following command to bring gazebo into action

  $ rosrun gazebo_ros gazebo –verbose
- Mostly, the above command won't work in the first go
- **Stop the process and Run again!** untill ..

Snapshot

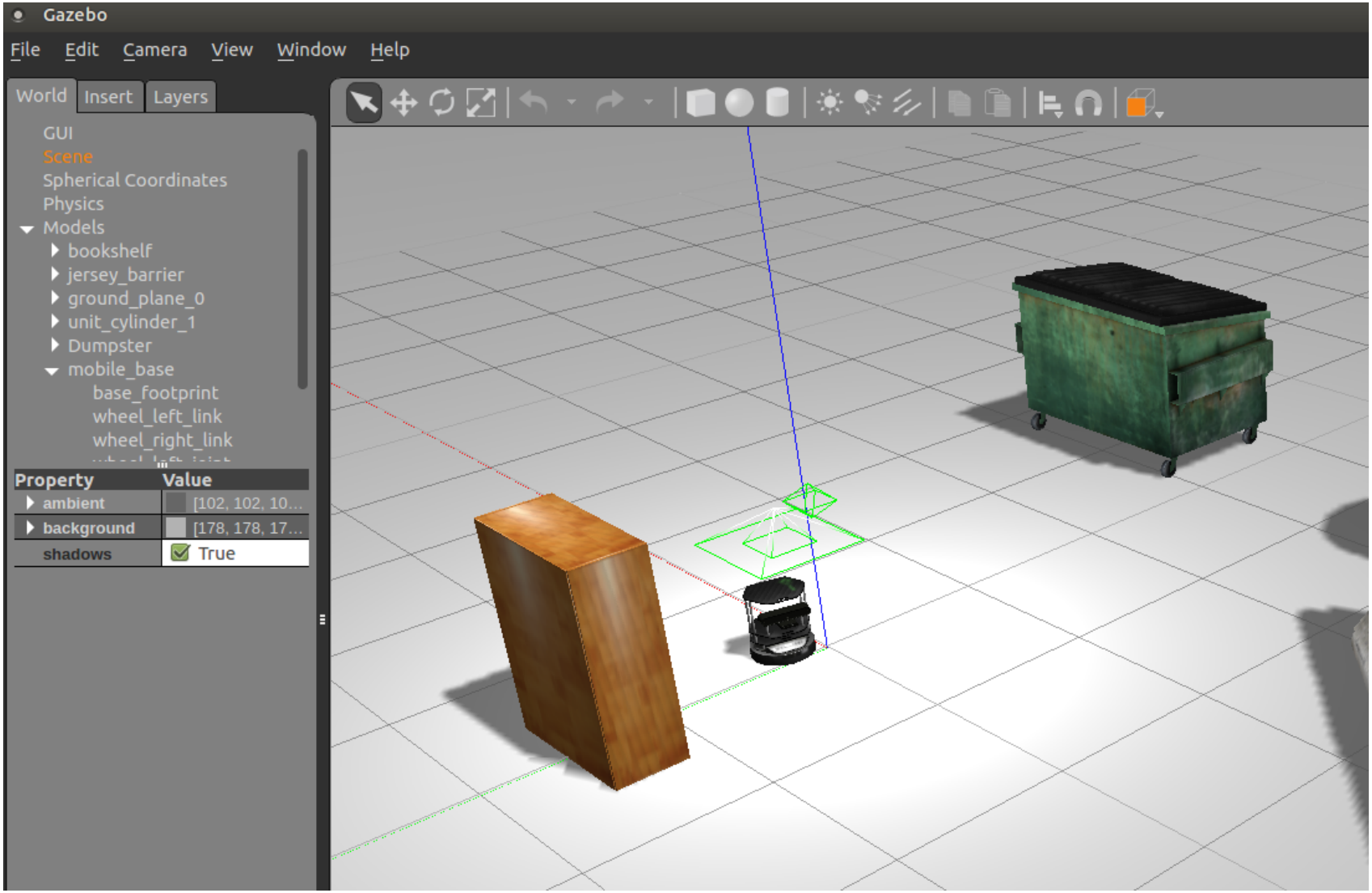

vivek@vivek-VirtualBox: ~/ros_files/catkin_ws
File Edit View Search Terminal Tabs Help
vivek@vivek-VirtualBox: ~/ros_fil...
roscore http://vivek-VirtualBox:11...
vivek@vivek-VirtualBox: ~/ros_fil...
vivek@vivek-VirtualBox:~/ros_files/catkin_ws$ rosrun gazebo_ros gazebo --verbose
Gazebo multi-robot simulator, version 7.0.0
Copyright (C) 2012-2016 Open Source Robotics Foundation.
Released under the Apache 2 License.
http://gazebosim.org
[INFO] [1579714106.146692986]: Finished loading Gazebo ROS API Plugin.
[Msg] Waiting for master.
[INFO] [1579714106.148207108]: waitForService: Service [/gazebo/set_physics_properties] has not been advertised, waiting...
[Msg] Connected to gazebo master @ http://127.0.0.1:11345
[Msg] Publicized address: 10.0.2.15
[INFO] [1579714106.632505605]: waitForService: Service [/gazebo/set_physics_properties] is now available.
[INFO] [1579714106.749042682]: Physics dynamic reconfigure ready.

Running the turtlebot

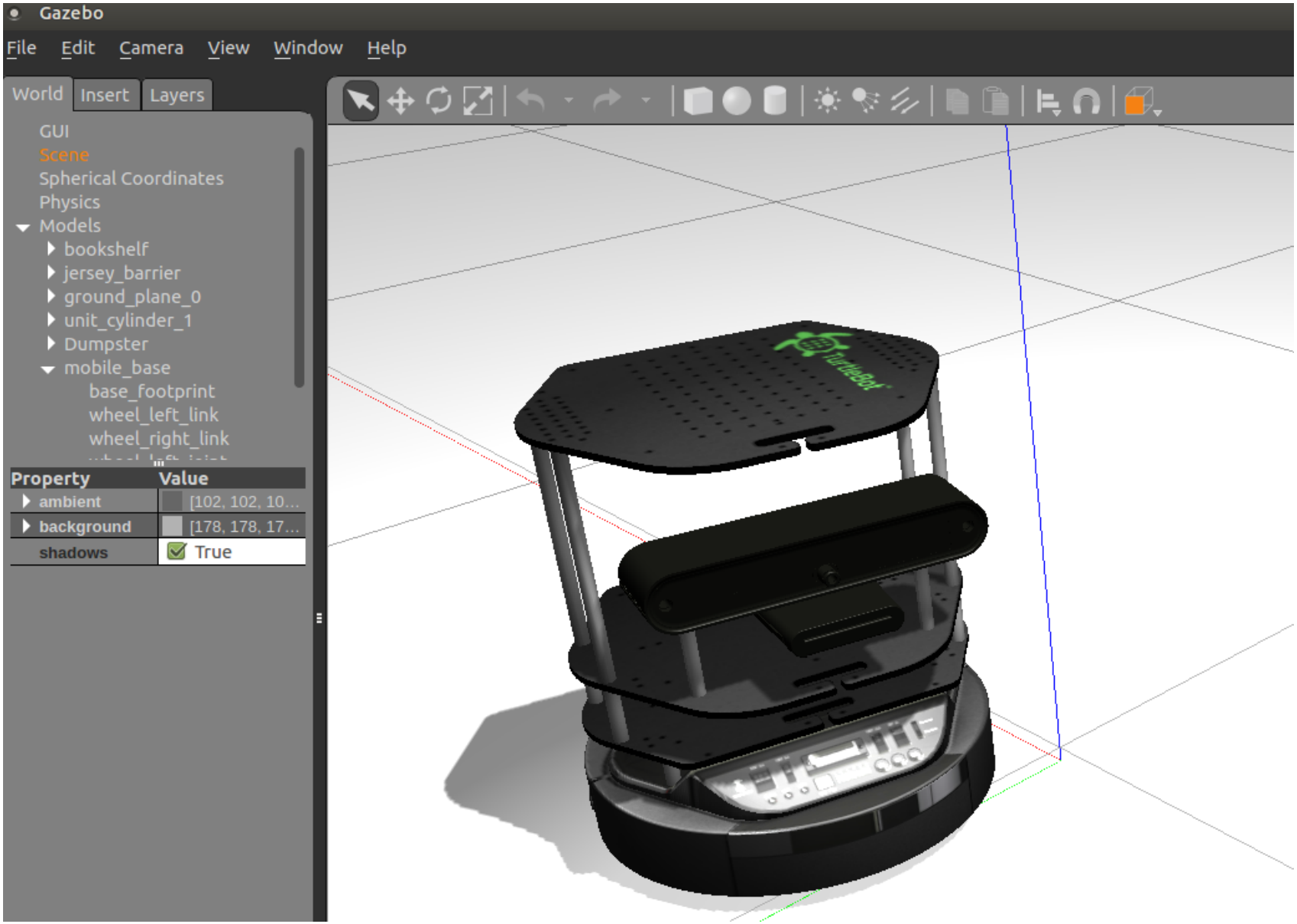
- Example gazebo environment with turtlebot

```
$ roslaunch turtlebot_gazebo turtlebot_world.launch
```

Snapshot


Gazebo
File
Edit
Camera
View
Window
Help
World
Insert
Layers
GUI
Scene
Spherical Coordinates
Physics
Models
bookshelf
jersey_barrier
ground_plane_0
unit_cylinder_1
Dumpster
mobile_base
base_footprint
wheel_left_link
wheel_right_link
Property
Value
ambient
[102, 102, 10...
background
[178, 178, 17...
shadows
True

Zoomed In


Gazebo
File
Edit
Camera
View
Window
Help
World
Insert
Layers
GUI
Scene
Spherical Coordinates
Physics
Models
bookshelf
jersey_barrier
ground_plane_0
unit_cylinder_1
Dumpster
mobile_base
base_footprint
wheel_left_link
wheel_right_link
Property
Value
ambient
[102, 102, 10...
background
[178, 178, 17...
shadows
True

Turtlebot topics

Launching the turtlebot_world simulation environment brings up varius nodes. Use command **$ rostopic list** to see.

Some of the nodes are

- ...
- /camera/rgb/camera_info
- /camera/rgb/image_raw
- ...
- /cmd_vel_mux/input/navi
- /cmd_vel_mux/input/teleop
- ...
- /odom
- /scan

Controlling turtlebot

Turtlebot can be controlled by publishing a *geometry_msgs/Twist* message to either of these topics:

- /cmd_vel_mux/input/navi
- /cmd_vel_mux/input/teleop

From commandline : **$ rostopic pub**

- rostopic pub -r 10 /cmd_vel_mux/input/navi geometry_msgs/Twist "linear:
  x: 0.3
  y: 0.0
  z: 0.0
  angular:
  x: 0.0
  y: 0.0
  z: 0.2"

Controlling turtlebot from script

Need to publish *geometry_msgs/Twist* messages.

```
1  #!/usr/bin/env python
2  import rospy
3  from geometry_msgs.msg import Twist
4
5  def control_loop():
6      pub = rospy.Publisher('/cmd_vel_mux/input/navi',
       Twist, queue_size=10)
7      rospy.init_node('turtlebot_controller')
8      rate = rospy.Rate(1)
9      velocity_msg = Twist()
10     velocity_msg.linear.x = 0.2
11     velocity_msg.angular.z = 0.2
12     while not rospy.is_shutdown():
13         pub.publish(velocity_msg)
14         rate.sleep()
15
16 if __name__ == '__main__':
17     try:
18         control_loop()
19     except rospy.ROSInterruptException:
20         pass
```

Getting position of turtlebot

The /odom topic provides *nav_msgs/Odometry* message

A single instance of the Odometry message looks like :

```
header:
  seq: 326038
  stamp:
    secs: 3260
    nsecs: 480000000
  frame_id: "odom"
child_frame_id: "base_footprint"
pose:
  pose:
    position:
      x: 0.206763016474
      y: -0.00754505523484
      z: 0.0
    orientation:
      x: 0.0
      y: -0.0
      z: 0.0549024539071
      w: -0.998491722827
  covariance: [...]
twist:
  twist:
    linear:
      x: -6.6322219357e-06
      y: 0.0
      z: 0.0
    angular:
      x: 0.0
      y: 0.0
      z: -5.59266047396e-05
  covariance: [...]
```

Quaternion to Euler angle

The orientation is encoded in **quaternion** format in the Odometry message.

Need to convert the quaternion into euler angles for computing control law.

The following snippet does the conversion:

```
1 from tf.transformations import euler_from_quaternion
2
3 def quat2euler(x,y,z,w):
4     quat = [x,y,z,w]
5     return euler_from_quaternion(quat)
```

A node to log the position data

```
1  #!/usr/bin/env python
2  import rospy
3  from nav_msgs.msg import Odometry
4  from quat2euler import quat2euler
5
6  def callback(data):
7      x  = data.pose.pose.orientation.x;
8      y  = data.pose.pose.orientation.y;
9      z = data.pose.pose.orientation.z;
10     w = data.pose.pose.orientation.w;
11     pose = [data.pose.pose.position.x, data.pose.pose.
       position.y, quat2euler(x,y,z,w)[2]]
12     print("Robot pose is : {:0.3f} {:0.3f} {:0.2f}".
       format(pose[0], pose[1], pose[2]))
13
14 def logger():
15     rospy.init_node('Log_odom')
16     rospy.Subscriber('/odom', Odometry, callback)
17     rospy.spin()
18
19 if __name__ == '__main__':
20     logger()
```

Project download

The project to log odometry data and control the *turtlebot* is available as a zip file at:

http://bit.ly/2tdpControllerBasic

Assignment

A) Write a python script to generate waypoints (x,y) and plot them:

$$x = A\ \cos(at)$$
$$y = A\ \sin(bt)$$

with A = 3, B = 3, a=1, b=2

B) Write a node that uses the waypoint generation code and use waypoints as target/goal point and the robots current location (from /odom) as start point. Calculate following error terms

$$E_{pos} : \text{Euclidean distance}$$
$$E_{\theta} : \text{Difference between current heading and desired heading}$$

C) Extend the node to apply proportional control taught in class to drive the robot along those waypoints. Plot the robot trajectory (superimposed onto waypoints)

Guidelines :
Comment your code. Keep separation between part A, B and C (use different functions with meaningful names OR use blank lines to create a visual separation)

Template code

A basic template for testing your code is available at
http://bit.ly/2tdpTemplateA2

Thank you