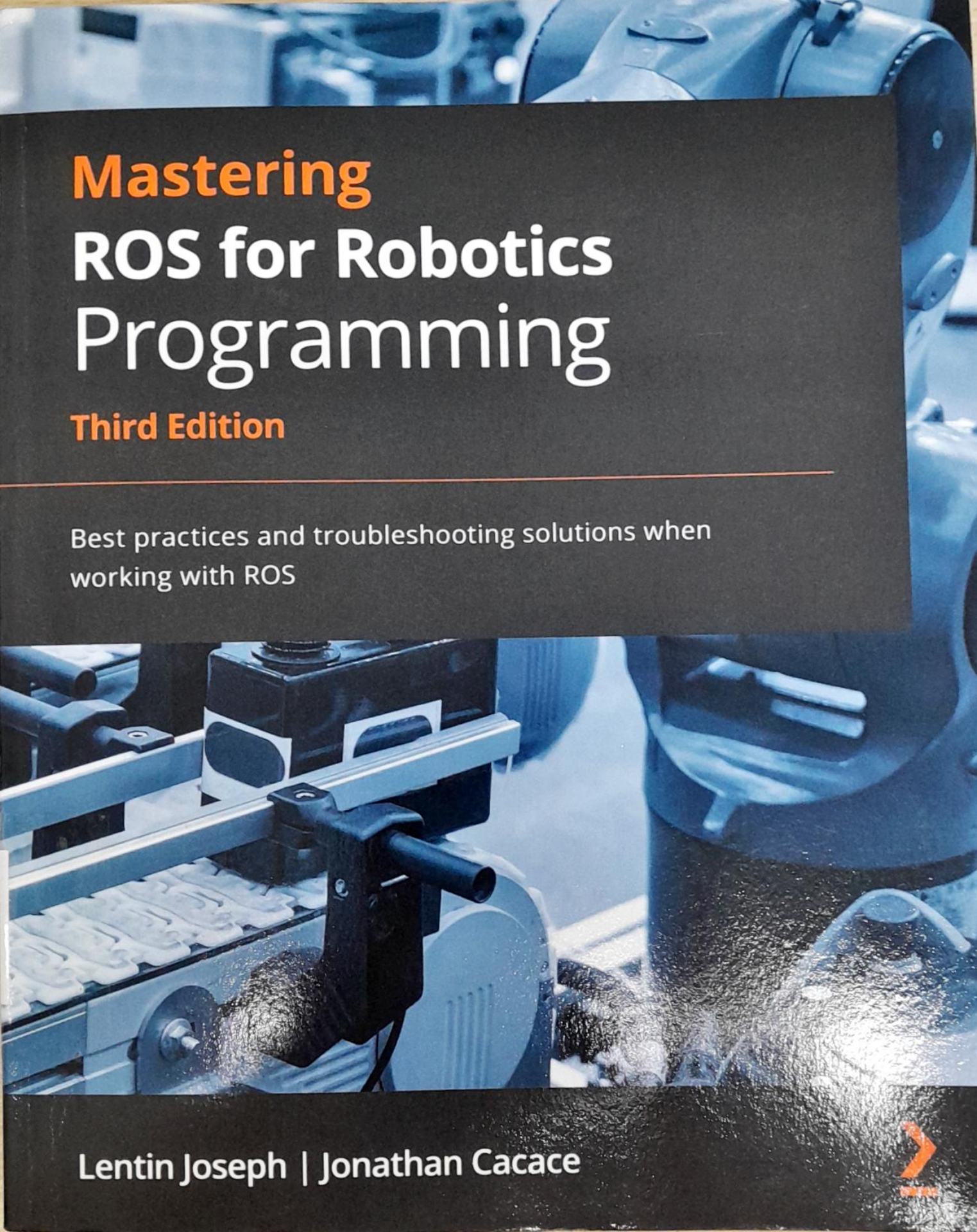


Mastering ROS for Robotics Programming

Third Edition

Best practices and troubleshooting solutions when working with ROS

Lentin Joseph | Jonathan Cacace

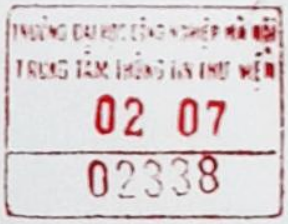
Mastering ROS for Robotics Programming

Third Edition

Best practices and troubleshooting solutions when working with ROS

Lentin Joseph

Jonathan Cacace



Packt>

BIRMINGHAM—MUMBAI

Mastering ROS for Robotics Programming

Third Edition

Group Product Manager: Wilson D'souza

Publishing Product Manager: Meeta Rajani

Senior Editor: Arun Nadar

Content Development Editor: Yasir Ali Khan

Technical Editor: Shruthi Shetty

Copy Editor: Safis Editing

Project Coordinator: Ajesh Devavaram

Proofreader: Safis Editing

Indexer: Manju Arasan

Production Designer: Alishon Mendonca

First published: December 2015

Second edition: February 2018

Third edition: September 2021

Production reference: 1230821

Published by Packt Publishing Ltd.

Livery Place

35 Livery Street

Birmingham

B3 2PB, UK.

ISBN 978-1-80107-102-4

www.packt.com

Contributors

About the authors

Lentin Joseph is an author, roboticist, and robotics entrepreneur from India. He runs a robotics software company called Qbotics Labs in Kochi, Kerala. He has 10 years of experience in the robotics domain, primarily with ROS, OpenCV, and PCL. He has authored other books on ROS, namely *Learning Robotics Using Python*, first and second edition, *Mastering ROS for Robotics Programming*, first and second edition, *ROS Robotics Projects*, first and second edition, and *ROS Learning Path and Robot Operating System for Absolute Beginners*. He pursued his master's in robotics and automation in India and also has worked at the Robotics Institute, CMU, USA. He is also a TEDx speaker.

I would like to dedicate this book to my parents (Jancy Joseph and CG Joseph) and my wife (Aleena Johny).

Jonathan Cacace was born in Naples, Italy, on December 13, 1987. He received a master's degree in computer science from the University of Naples Federico II in 2012 and a Ph.D. degree in robotics in 2016 from the same institution. Currently, he is an assistant professor at the **PRISMA Lab** (**Projects of Robotics for Industry and Services, Mechatronics and Automation Laboratory**) at the University of Naples Federico II, where he is involved in several research projects in the fields of human-robot interaction in industry 4.0 and the autonomous control of UAVs for inspection, maintenance, and robotic manipulation.

I would like to dedicate this book to my family.

About the reviewers

Nick Rotella earned his B.Sc. degree in mechanical engineering from the Cooper Union, followed by his M.Sc. and Ph.D. degrees in computer science from the University of Southern California. As a roboticist, Nick considers himself to be a well-rounded scientist, developer, and engineer. While his Ph.D. thesis focused heavily on model-based motion planning and controls for humanoid robots, he has also worked on autonomous applications in the marine, drone, automotive, mining, and logistics spaces. His experience in controls is based on a deep theoretical understanding of dynamics, estimation, and trajectory generation; Nick has written software at all levels of autonomous system stacks for high-performance controls.

Prateek Nagras is the founder of TechnoYantra (`https://www.technoyantra.com/`), a service-based robotics start-up based in Pune, India.

He is an engineer who studied instrumentation and control engineering at VIT, Pune, and mechatronics with a specialization in robotics at FH Aachen in Germany.

Having gained valuable experience as a robotics engineer in Germany and Austria, he decided to come back to India and started TechnoYantra in December 2019.

TechnoYantra specializes in providing custom robotic solutions to clients in the automobile, health, industrial, and agricultural sectors in the US, Germany, the Netherlands, Saudi Arabia, Singapore, and more.

When Prateek is not building robots, you can find him playing football or watching sports.

Table of Contents

Preface

Section 1 – ROS Programming Essentials

1
Introduction to ROS

Technical requirements 3
Why should we use ROS? 4
Understanding the ROS filesystem level 5
ROS packages 7
ROS metapackages 9
ROS messages 10
The ROS services 12
Understanding the ROS computation graph level 13
ROS nodes 15
ROS messages 16
ROS topics 16
ROS services 17
ROS bagfiles 18
The ROS master 19
Using the ROS parameter 20
ROS community level 22
Prerequisites for starting with ROS 22
ROS distributions 23
Running the ROS master and the ROS parameter server 24
Summary 27
Questions 27

2
Getting Started with ROS Programming

Technical requirements 30
Creating a ROS package 30
Working with ROS topics 33
Creating ROS nodes 33
Building the nodes 36
Adding custom .msg and .srv files 38
Working with ROS services 42
Working with ROS actionlib 46
Building the ROS action server and client 51

Creating launch files 53
Applications of topics, services, and actionlib 56
Summary 56
Questions 57

Section 2 – ROS Robot Simulation

3

Working with ROS for 3D Modeling

Technical requirements 62
ROS packages for robot modeling 63
Understanding robot modeling using URDF 64
Creating the ROS package for the robot description 68
Creating our first URDF model 68
Explaining the URDF file 71
Visualizing the 3D robot model in RViz 73
Interacting with pan-and-tilt joints 74
Adding physical and collision properties to a URDF model 76
Understanding robot modeling using xacro 77
Using properties 78
Using the math expression 78
Converting xacro to URDF 79
Creating the robot description for a seven-DOF robot manipulator 80
Arm specification 81
Explaining the xacro model of the seven-DOF arm 82
Using constants 82
Using macro 83
Including other xacro files 84
Using meshes in the link 84
Working with the robot gripper 85
Viewing the seven-DOF arm in RViz 86
Creating a robot model for the differential drive mobile robot 89
Summary 95
Questions 95

4

Simulating Robots Using ROS and Gazebo

Technical requirements 98
Simulating the robotic arm using Gazebo and ROS 98
Creating the robotic arm simulation model for Gazebo 99
Adding colors and textures to the Gazebo robot model 101
Adding transmission tags to actuate the model 102
Adding the gazebo_ros_control plugin 102

Adding a 3D vision sensor to Gazebo 103

Simulating the robotic arm with Xtion Pro 105

Visualizing the 3D sensor data 106

Moving the robot joints using ROS controllers in Gazebo 107

Understanding the ros_control packages 108

Different types of ROS controllers and hardware interfaces 108

How the ROS controller interacts with Gazebo 109

Interfacing the joint state controllers and joint position controllers with the arm 110

Launching the ROS controllers with Gazebo 112

Moving the robot joints 114

Simulating a differential wheeled robot in Gazebo 115

Adding the laser scanner to Gazebo 117

Moving the mobile robot in Gazebo 119

Adding joint state publishers to the launch file 121

Adding the ROS teleop node 121

Questions 123

Summary 124

5

Simulating Robots Using ROS, CoppeliaSim, and Webots

Technical requirements 126

Setting up CoppeliaSim with ROS 126

Understanding the RosInterface plugin 130

Working with ROS messages 134

Simulating a robotic arm using CoppeliaSim and ROS 136

Adding the ROS interface to CoppeliaSim joint controllers 138

Setting up Webots with ROS 140

Introduction to the Webots simulator 141

Simulating a mobile robot with Webots 142

Writing your first controller 145

Simulating the robotic arm using Webots and ROS 148

Writing a teleop node using webots_ros 149

Starting Webots with a launch file 153

Summary 154

Questions 154

6

Using the ROS MoveIt! and Navigation Stack

Technical requirements 156

The MoveIt! architecture 157

The move_group node 158

Motion planning using MoveIt! 159

Motion-planning request adapters 160
MoveIt! planning scene 161
MoveIt! kinematics handling 162
MoveIt! collision checking 162

Generating a MoveIt! configuration package using the Setup Assistant tool 163
Step 1 – Launching the Setup Assistant tool 163
Step 2 – Generating a self-collision matrix 166
Step 3 – Adding virtual joints 166
Step 4 – Adding planning groups 167
Step 5 – Adding the robot poses 168
Step 6 – Setting up the robot end effector 169
Step 7 – Adding passive joints 170
Step 8 – Author information 170
Step 9 – Generating configuration files 170

Motion planning of a robot in RViz using the MoveIt! configuration package 171
Using the RViz MotionPlanning plugin 172
Interfacing the MoveIt! configuration package to Gazebo 176

Understanding the ROS Navigation stack 183
ROS Navigation hardware requirements 183
Working with Navigation packages 185
Workings of the Navigation stack 187

Building a map using SLAM 188
Creating a launch file for gmapping 189
Running SLAM on the differential drive robot 191
Implementing autonomous navigation using amcl and a static map 194
Creating an amcl launch file 195

Summary 198
Questions 198

7

Exploring the Advanced Capabilities of ROS MoveIt!

Technical requirements 200

Motion planning using the move_group C++ interface 200
Motion planning a random path using MoveIt! C++ APIs 201
Motion planning a custom path using MoveIt! C++ APIs 202
Collision checking with a robot arm using MoveIt! 205

Working with perception using MoveIt! and Gazebo 213

Performing object manipulation with MoveIt! 219
Working with a robot pick-and-place task using MoveIt! 220
Pick-and-place actions in Gazebo and real robots 224

Understanding DYNAMIXEL ROS servo controllers for robot hardware interfacing 225
DYNAMIXEL servos 225
DYNAMIXEL-ROS interface 226

Interfacing a 7-DOF DYNAMIXEL-based robotic arm with ROS MoveIt! 227
Creating a controller package for a COOL arm robot 228
MoveIt! configuration of the COOL arm 233
Summary 235
Questions 235

8
ROS for Aerial Robots

Technical requirements 238
Using aerial robots 238
UAV hardware 239
Pixhawk autopilot 240
Using the PX4 flight control stack 242
PX4 firmware architecture 246
PX4 SITL 248
PC/autopilot communication 249
The mavros ROS package 252
Writing a ROS-PX4 application 252
Writing a trajectory streamer 259
External pose estimation for PX4 261
Using the RotorS simulation framework 262
Installing RotorS 263
RotorS packages 265
Creating a new UAV model 267
Interacting with RotorS motor models 277
Summary 279
Questions 279

Section 3 – ROS Robot Hardware Prototyping

9
Interfacing I/O Board Sensors and Actuators to ROS

Technical requirements: 284
Understanding the Arduino-ROS interface 284
What is the Arduino-ROS interface? 285
Understanding the rosserial package in ROS 286
Understanding ROS node APIs in Arduino 292
ROS-Arduino Publisher and Subscriber example 294
Arduino-ROS example – blinking an LED with a push button 298
Arduino-ROS example – Accelerometer ADXL 335 301

Arduino-ROS example – ultrasonic distance sensor 303
Arduino-ROS example – odometry data publisher 307

Interfacing non-Arduino boards to ROS 309

Setting up the Odroid-C4, Raspberry Pi 4, and Jetson Nano for installing ROS 309
Blinking the LED using ROS on the Raspberry Pi 4 318
A push button and a blinking LED using ROS on the Raspberry Pi 2 321
Running examples on the Raspberry Pi 4 325

Interfacing DYNAMIXEL actuators to ROS 326

Summary 327

Questions 327

10

Programming Vision Sensors Using ROS, OpenCV, and PCL

Technical requirements 330

Understanding ROS – OpenCV interfacing packages 330

Understanding ROS – PCL interfacing packages 331

Installing ROS perception 332

Interfacing USB webcams in ROS 334

Working with ROS camera calibration 338

Converting images between ROS and OpenCV using cv_bridge 341

Interfacing Kinect and Asus Xtion Pro with ROS 348

Interfacing the Intel RealSense camera with ROS 351

Converting point cloud to a laser scan 353

Interfacing Hokuyo lasers with ROS 356

Interfacing RPLIDAR and YDLIDAR with ROS 358

Working with point cloud data 358

How to publish a point cloud 359
How to subscribe and process a point cloud 361
Reading and publishing a point cloud from a PCD file 364

Summary 368

Questions 368

11

Building and Interfacing Differential Drive Mobile Robot Hardware in ROS

Technical requirements 370

Software requirements 370
Network setup 372
Hardware requirements 373

Introduction to the Remo robot – a DIY autonomous mobile robot 373
Remo hardware components 374
Software requirements for the ROS Navigation Stack 377
Developing a low-level controller and a high-level ROS Control hardware interface for a differential drive robot 378
Implementing the low-level base controller for Remo 380
ROS Control high-level hardware interface for a differential drive robot 386
Overview of ROS nodes and topics for the Remo robot 389
Configuring and working with the Navigation Stack 393
Configuring the gmapping node and creating a map 393
Working with the gmapping node 393
Configuring the move_base node 395
Configuring the AMCL node 397
AMCL planning 397
Working with Remo robot in simulation 400
Summary 402
Questions 402

Section 4 – Advanced ROS Programming

12
Working with pluginlib, nodelets, and Gazebo Plugins

Technical requirements 406
Understanding pluginlib 406
Implementing a calculator plugin using pluginlib 407
Understanding ROS nodelets 414
Implementing a sample nodelet 414
Understanding and creating a Gazebo plugin 421
Creating a basic world plugin 422
Summary 427
Questions 427

13
Writing ROS Controllers and Visualization Plugins

Technical requirements 430
Understanding ros_control packages 430
The controller_interface package 431
Writing a basic joint controller in ROS 435
Step 1 - creating the controller package 435
Step 2 - creating the controller header file 436
Step 3 - creating the controller source file 437
Step 4 - detailed explanation of the controller source file 439

Step 5 – creating the plugin description file 440
Step 6 – updating package.xml 441
Step 7 – updating CMakeLists.txt 441
Step 8 – building the controller 441
Step 9 – writing the controller configuration file 442
Step 10 – writing the launch file for the controller 442
Step 11 – running the controller along with the seven-DOF arm in Gazebo 443

Understanding the RViz tool and its plugins 446

The Displays panel 447
The RViz toolbar 447
The Views panel 448
The Time panel 448
Dockable panels 448

Writing an RViz plugin for teleoperation 448

The methodology of building a RViz plugin 449

Summary 457
Questions 457

14

Using ROS in MATLAB and Simulink

Technical requirements 460
Getting started with MATLAB 460
Getting started with ROS Toolbox and MATLAB 462
Starting with ROS topics and MATLAB callback functions 466

Developing a robotic application using MATLAB and Gazebo 470
Getting started with ROS and Simulink 474

Creating a wave signal integrator in Simulink 474
Publishing a ROS message in Simulink 477
Subscribing to a ROS topic in Simulink 481

Developing a simple control system in Simulink 483
Configuring the Simulink model 486

Summary 487
Questions 488

15

ROS for Industrial Robots

Technical requirements 490
Understanding ROS-Industrial packages 490
Goals of ROS-Industrial 490
ROS-Industrial – a brief history 491

Installing ROS-Industrial packages 491
Block diagram of ROS-Industrial packages 492

Creating a URDF for an industrial robot 494
Creating the MoveIt configuration for an industrial robot 496
Updating the MoveIt configuration files 500
Installing ROS-Industrial packages for Universal Robots arms 502
Installing the ROS interface for Universal Robots 503
Understanding the MoveIt configuration of a Universal Robots arm 505
Getting started with real Universal Robots hardware and ROS-I 508
Working with MoveIt configuration for ABB robots 510
Understanding the ROS-Industrial robot support packages 513
The ROS-Industrial robot client package 515
Designing industrial robot client nodes 515
The ROS-Industrial robot driver package 517
Understanding the MoveIt IKFast plugin 520
Creating the MoveIt IKFast plugin for the ABB IRB 6640 robot 520
Prerequisites for developing the MoveIt IKFast plugin 520
The OpenRave and IKFast modules 521
MoveIt IKFast 521
Installing the MoveIt IKFast package 521
Installing OpenRave on Ubuntu 20.04 522
Creating the COLLADA file of a robot to work with OpenRave 523
Generating the IKFast CPP file for the IRB 6640 robot 525
Creating the MoveIt IKFast plugin 526
Summary 528
Questions 528

16

Troubleshooting and Best Practices in ROS

Setting up Visual Studio Code with ROS 530
Installing/uninstalling Visual Studio Code 531
Getting started with Visual Studio Code 531
Installing new Visual Studio Code extensions 533
Getting started with the Visual Studio Code ROS extension 534
Inspecting and building the ROS workspace 536
Managing ROS packages using Visual Studio Code 537
Visualizing the preview of a URDF file 538
Best practices in ROS 539
ROS C++ coding style guide 539

Best coding practices for the ROS package 543
Important troubleshooting tips in ROS 544
Using roswtf 544
Summary 547
Questions 547

Other Books You May Enjoy

Index

Preface

The **Robot Operating System** (**ROS**) is a globally used robotics middleware that helps developers to program robotic applications and is currently adopted by robotics companies, research centers, and universities to program advanced robots. *Mastering ROS for Robotics Programming, Third Edition* presents advanced concepts of the ROS framework and is particularly suitable for users who are already familiar with the basic concepts of ROS. However, a brief introduction to the basic ROS concepts is provided in the first chapter in order to help new developers start with the examples in the book.

You will be guided through the creation, the modeling and design, of new robots, as well as simulating and interfacing them with the ROS framework. You will use advanced simulation software to use ROS tools that allow robot navigation, manipulation, and sensor elaboration. Finally, you will learn how to handle important concepts such as ROS low-level controllers, nodelets, and plugins.

You can work with almost all of the examples of the book using only a standard computer without any special hardware requirements. However, additional hardware components will be used in some chapters of the book to discuss how to use ROS with external sensors, actuators, and I/O boards.

The book is organized as follows: after an introduction to the basic concepts of ROS, how to model and simulate a robot is discussed. Gazebo, CoppeliaSim, and the Webots software simulator will be used to control and interact with the modeled robot. These simulators will be used to connect to robots with the MoveIt! and navigation ROS packages. ROS plugins, controllers, and nodelets are then discussed. Finally, the book discusses how to connect MATLAB and Simulink with ROS.

Who this book is for

This book is meant to be used by passionate robotics developers or researchers who want to fully exploit the features of ROS. The book is also good for users who are already familiar with typical robotics applications or who want to start learning how to develop in the world of ROS in an advanced manner, learning how to model, build, and control their robots. Basic knowledge of GNU/Linux and C++ programming is strongly recommended if you want to easily comprehend the contents of the book.

What this book covers

Chapter 1, Introduction to ROS, gives you an understanding of the core underlying concepts of ROS.

Chapter 2, Getting Started with ROS Programming, explains how to work with ROS packages.

Chapter 3, Working with ROS for 3D Modeling, discusses the design of two robots; one is a seven **Degrees of Freedom** (**DOF**) manipulator, and the other is a differential drive robot.

Chapter 4, Simulating Robots Using ROS and Gazebo, discusses the simulation of a seven-DOF arm, differential wheeled robots, and ROS controllers that help control robot joints in Gazebo.

Chapter 5, Simulating Robots Using ROS, CoppeliaSim and Webots, introduces the CoppeliaSim and Webots simulators, showing how to simulate and control different types of robots.

Chapter 6, Using the ROS MoveIt! and Navigation Stack On, covers out-of-the-box functionalities such as robot manipulation and autonomous navigation using ROS MoveIt! and the navigation stack.

Chapter 7, Exploring the Advanced Capabilities of ROS-MoveIt!, discusses the capabilities of MoveIt!, such as collision avoidance, perception using 3D sensors, grasping, picking, and placing. After that, we will see how to interface robotic manipulator hardware with MoveIt!.

Chapter 8, ROS for Aerial Robots, discusses how to simulate and control aerial robots with ROS, considering the particular case of quadcopters.

Chapter 9, Interfacing I/O Boards, Sensors, and Actuators to ROS, discusses interfacing some hardware components, such as sensors and actuators, with ROS. We will look at the interfacing of sensors using I/O boards, such as Arduino or Raspberry Pi, with ROS.

Chapter 10, Programming Vision Sensors Using ROS, OpenCV, and PCL, discusses how to interface various vision sensors with ROS and program them using libraries such as **Open Source Computer Vision** (**OpenCV**) and **Point Cloud Library** (**PCL**).

Chapter 11, Building and Interfacing Differential Drive Mobile Robot Hardware in ROS, helps you to build autonomous mobile robot hardware with differential drive configuration and interface it with ROS. This chapter aims to give you an understanding of building a custom mobile robot and interfacing it with ROS.

Chapter 12, Working with pluginlib, Nodelets, and Gazebo Plugins, shows some of the advanced concepts in ROS, such as ROS `pluginlib`, `nodelets`, and `Gazebo` plugins. We will discuss the functionalities and application of each concept and will practice one example to demonstrate their workings.

Chapter 13, Writing ROS Controllers and Visualization Plugins, shows how to write and run a basic ROS controller. We will also see how to create a plugin for RViz.

Chapter 14, Using ROS in MATLAB and Simulink, discusses how to connect MATLAB and Simulink with ROS.

Chapter 15, ROS for Industrial Robots, helps you understand and install ROS-Industrial packages in ROS. We will see how to develop a MoveIt! IKFast plugin for an industrial robot.

Chapter 16, Troubleshooting and Best Practices in ROS, discusses how to set up a ROS development environment in Eclipse IDE, best practices in ROS, and troubleshooting tips in ROS.