

ROS Robotics Projects

Second Edition

Build and control robots powered by the Robot Operating System,
machine learning, and virtual reality



Packt>

www.packt.com

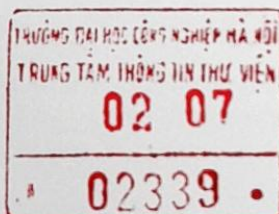
Ramkumar Gandhinathan and Lentin Joseph

ROS Robotics Projects

Second Edition

Build and control robots powered by the Robot Operating System, machine learning, and virtual reality

Ramkumar Gandhinathan
Lentin Joseph



Packt

BIRMINGHAM - MUMBAI

ROS Robotics Projects

Second Edition

Copyright © 2019 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the authors, nor Packt Publishing or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

Commissioning Editor: Vijin Boricha
Acquisition Editor: Meeta Rajani
Content Development Editor: Pratik Andrade
Senior Editor: Rahul Dsouza
Technical Editor: Dinesh Pawar
Copy Editor: Safis Editing
Project Coordinator: Anish Daniel
Proofreader: Safis Editing
Indexer: Rekha Nair
Production Designer: Alishon Mendonsa

First published: March 2017
Second edition: December 2019

Production reference: 1181219

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham
B3 2PB, UK.

ISBN 978-1-83864-932-6

www.packt.com

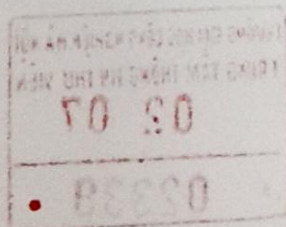


Table of Contents

Preface	1
Chapter 1: Getting Started with ROS	7
Technical requirements	8
Getting started with ROS	8
ROS distributions	9
Supported OSES	10
Robots and sensors supported by ROS	11
Why use ROS?	13
Fundamentals of ROS	14
The filesystem level	15
The computation graph level	16
The ROS community level	18
Communication in ROS	19
ROS client libraries	20
ROS tools	21
ROS Visualizer (RViz)	21
rqt_plot	22
rqt_graph	23
ROS simulators	24
Installing ROS Melodic on Ubuntu 18.04 LTS	25
Getting started with the installation	26
Configuring Ubuntu repositories	26
Setting up source.list	28
Setting up keys	29
Installing ROS Melodic	29
Initializing rosdep	29
Setting up the ROS environment	30
Getting rosinstall	30
Setting up ROS on VirtualBox	31
Introduction to Docker	33
Why Docker?	34
Installing Docker	35
Installing from the Ubuntu repository	35
Removing Docker	35
Installing from the Docker repository	36
Working with Docker	37
Setting up the ROS workspace	39
Opportunities for ROS in industries and research	41
Summary	41

Chapter 2: Introduction to ROS-2 and Its Capabilities	43
Technical requirements	45
Getting started with ROS-2	45
ROS-2 distributions	47
Supported operating systems	47
Robots and sensors supported in ROS-2	48
Why ROS-2?	48
Fundamentals of ROS-2	49
What is DDS?	50
How is DDS implemented?	50
Computational graph	51
ROS-2 community level	51
Communication in ROS-2	52
Changes between ROS-1 and ROS-2	53
ROS-2 client libraries (RCL)	54
ROS-2 tools	55
Rviz2	56
Rqt	57
Installing ROS-2	58
Getting started with the installation	58
Setting up the system locale	58
Adding ROS-2 repositories	59
Installing development and ROS tools	60
Getting the ROS-2 source code	60
Installing dependencies using rosdep	61
Installing DDS implementations (optional)	62
Building code	63
Setting up ROS-1, ROS-2, or both environments	64
Running test nodes	66
Setting up the ROS-2 workspace	68
Writing ROS-2 nodes	69
ROS-1 example code	70
ROS-2 example code	71
Differences between ROS-1 and ROS-2 talker nodes	75
Bridging ROS-1 and ROS-2	76
Testing the ros1_bridge package	77
Summary	79
Chapter 3: Building an Industrial Mobile Manipulator	81
Technical requirements	82
Understanding available mobile manipulators	83
Applications of mobile manipulators	85
Getting started building mobile manipulators	86
Units and coordinate system	87
Gazebo and ROS assumptions	87

Building the robot base	89
Robot base prerequisites	89
Robot base specifications	89
Robot base kinematics	90
Software parameters	91
ROS message format	91
ROS controllers	92
Modeling the robot base	92
Initializing the workspace	94
Defining the links	95
Defining the joints	97
Simulating the robot base	98
Defining collisions	98
Defining actuators	99
Defining ROS_CONTROLLERS	100
Testing the robot base	102
Getting started building the robot arm	106
Robot arm prerequisites	106
Robot arm specifications	106
Robot arm kinematics	107
Software parameters	108
The ROS message format	108
ROS controllers	108
Modeling the robot arm	109
Initializing the workspace	109
Defining the links	110
Defining the joints	111
Simulating the robot arm	112
Defining collisions	112
Defining actuators	113
Defining ROS_CONTROLLERS	114
Testing the robot arm	115
Putting things together	117
Modeling the mobile manipulator	117
Simulating and testing the mobile manipulator	118
Summary	119
Chapter 4: Handling Complex Robot Tasks Using State Machines	121
Technical requirements	122
Introduction to ROS actions	122
The client-server concept	123
An actionlib example – robot arm client	124
An actionlib example – battery simulator server-client	127
Creating a package and a folder action inside it	128
Creating an action file that has the goal, result, and feedback	128
Modifying the package files and compiling the package	129
Defining a server	130
Defining a client	132

Waiter robot analogy	134
Introduction to state machines	136
Introduction to SMACH	138
SMACH concepts	138
Outcome	140
User data	140
Preemption	140
Introspection	140
Getting started with SMACH examples	142
Installing and using SMACH-ROS	142
Simple example	142
Restaurant robot analogy	146
Summary	150
Chapter 5: Building an Industrial Application	151
Technical requirements	152
Application use case – robot home delivery	152
Setting up the environment in Gazebo	155
Making our robot base intelligent	156
Adding a laser sensor	156
Configuring the navigation stack	159
Mapping the environment	161
Localizing the robot base	162
Making our robot arm intelligent	163
Introduction to Moveit	163
Installing and configuring Moveit for our mobile robot	165
Installing Moveit	165
Configuring the Moveit setup assistant wizard	165
Loading the robot model	166
Setting up self-collisions	167
Setting up planning groups	167
Setting up arm poses	168
Setting up passive joints	169
Setting up ROS controllers	170
Finalizing the Moveitconfig package	170
Controlling the robot arm using Moveit	171
Simulating the application	175
Mapping and saving the environment	175
Choosing the points on the environment	176
Adding the points to our library	177
Completing the state machine	177
Improvements to the robot	177
Summary	178
Chapter 6: Multi-Robot Collaboration	179
Technical requirements	180

Understanding the swarm robotics application	180
Swarm robot classification	181
Multiple robot communication in ROS	183
Single roscore and common networks	183
Issues with a common network	185
Using groups/namespaces	186
Example – multi-robot spawn using groups/namespaces	187
Issues with using groups/namespaces	190
Introduction to the multimaster concept	191
Introduction to the multimaster_fkie package	192
Installing the multimaster_fkie package	193
Setting up the multimaster_fkie package	193
Setting up hostnames and IPs	194
Checking and enabling the multicast feature	195
Testing the setup	195
A multi-robot use case	197
Summary	199
Chapter 7: ROS on Embedded Platforms and Their Control	201
Technical requirements	202
Understanding embedded boards	202
Important concepts	204
How different are microcontrollers and microprocessors in robotics?	205
What matters while choosing such boards	205
Introduction to microcontroller boards	206
Arduino Mega	206
How to choose an Arduino board for your robot	207
STM32	208
ESP8266	209
ROS-supported embedded boards	210
OpenCR	210
Arbotix-Pro	211
Comparison table	211
Introduction to single-board computers	212
CPU boards	213
Tinkerboard S	213
BeagleBone Black	214
Raspberry Pi	215
Comparison table	216
GPU boards	217
Jetson TX2	217
Jetson Nano	218
Comparison table	218
Debian versus Ubuntu	219
Setting up ROS on Tinkerboard S	220
Prerequisites	220

Installing the Tinkerboard Debian OS	220
Installing Armbian and ROS	222
Installing using an available ROS image	224
Setting up ROS on BeagleBone Black	225
Prerequisites	225
Installing the Debian OS	225
Installing Ubuntu and ROS	226
Setting up ROS on Raspberry Pi 3/4	228
Prerequisites	228
Installing Raspbian and ROS	229
Installing Ubuntu and ROS	230
Setting up ROS on Jetson Nano	231
Controlling GPIOs from ROS	232
Tinkerboard S	232
BeagleBone Black	233
Raspberry Pi 3/4	235
Jetson Nano	236
Benchmarking embedded boards	237
Getting started with Alexa and connecting with ROS	240
Alexa skill-building requirements	240
Creating a skill	242
Summary	248
Chapter 8: Reinforcement Learning and Robotics	249
Technical requirements	250
Introduction to machine learning	250
Supervised learning	251
Unsupervised learning	251
Reinforcement learning	252
Understanding reinforcement learning	252
Explore versus exploit	253
Reinforcement learning formula	254
Reinforcement learning platforms	256
Reinforcement learning in robotics	257
MDP and the Bellman equation	257
Reinforcement learning algorithms	260
Taxi problem analogy	260
TD prediction	261
Algorithm explanation	261
TD control	263
Off-policy learning – the Q-learning algorithm	263
Algorithm explanation	264
On-policy learning – the SARSA algorithm	267
Algorithm explanation	268
Installing OpenAI Gym, NumPy, and pandas	270
Q-learning and SARSA in action	271

Reinforcement learning in ROS	272
gym-gazebo	272
TurtleBot and its environment	274
Installing gym-gazebo and its dependencies	276
Testing the TurtleBot-2 environment	277
gym-gazebo2	280
MARA and its environment	280
Installing gym-gazebo2 and dependencies	281
Testing the MARA environment	283
Summary	284
Chapter 9: Deep Learning Using ROS and TensorFlow	285
Technical requirements	286
Introduction to deep learning and its applications	286
Deep learning for robotics	287
Deep learning libraries	288
Getting started with TensorFlow	289
Installing TensorFlow on Ubuntu 18.04 LTS	289
TensorFlow concepts	292
Graph	292
Session	293
Variables	293
Fetches	294
Feeds	294
Writing our first code in TensorFlow	295
Image recognition using ROS and TensorFlow	298
Prerequisites	298
The ROS image recognition node	299
Running the ROS image recognition node	301
Introducing to scikit-learn	304
Installing scikit-learn on Ubuntu 18.04 LTS	304
Introduction to SVM and its application in robotics	305
Implementing an SVM-ROS application	305
Summary	308
Chapter 10: Creating a Self-Driving Car Using ROS	309
Technical requirements	310
Getting started with self-driving cars	310
The history of autonomous vehicles	310
Levels of autonomy	312
Components of a typical self-driving car	313
GPS, IMU, and wheel encoders	313
Xsens MTi IMU	314
Camera	314
Ultrasonic sensors	314
LIDAR and RADAR	315
Velodyne HDL-64 LIDAR	315

SICK LMS 5xx/1xx and Hokuyo LIDAR	316
Continental ARS 300 radar (ARS)	317
The Delphi radar	317
Onboard computer	317
Software block diagram of self-driving cars	318
Simulating and interfacing self-driving car sensors in ROS	319
Simulating the Velodyne LIDAR	320
Interfacing Velodyne sensors with ROS	322
Simulating a laser scanner	323
Explaining the simulation code	326
Interfacing laser scanners with ROS	327
Simulating stereo and mono cameras in Gazebo	328
Interfacing cameras with ROS	331
Simulating GPS in Gazebo	331
Interfacing GPS with ROS	333
Simulating IMU on Gazebo	333
Interfacing IMUs with ROS	336
Simulating an ultrasonic sensor in Gazebo	336
Low-cost LIDAR sensors	338
Sweep LIDAR	338
RPLIDAR	340
Simulating a self-driving car with sensors in Gazebo	341
Installing prerequisites	341
Visualizing robotic car sensor data	344
Moving a self-driving car in Gazebo	344
Running hector SLAM using a robotic car	345
Interfacing a DBW car with ROS	347
Installing packages	347
Visualizing the self-driving car and sensor data	347
Communicating with DBW from ROS	349
Introducing the Udacity open source self-driving car project	350
Open source self-driving car simulator from Udacity	351
MATLAB ADAS Toolbox	354
Summary	355
Chapter 11: Teleoperating Robots Using a VR Headset and Leap Motion	
Technical requirements	357
Getting started with a VR headset and Leap Motion	358
Designing and working on the project	359
Installing the Leap Motion SDK on Ubuntu 14.04.5	362
Visualizing the Leap Motion controller data	363
Playing with the Leap Motion Visualizer tool	364
Installing the ROS driver for the Leap Motion controller	365
Testing the Leap Motion ROS driver	367
Visualizing Leap Motion data in RViz	368
	370

Creating a teleoperation node using the Leap Motion controller	372
Building a ROS-VR Android application	374
Working with the ROS-VR application and interfacing with Gazebo	376
TurtleBot simulation in VR	379
Installing the Turtlebot simulator	380
Working with TurtleBot in VR	381
Troubleshooting the ROS-VR application	382
Integrating the ROS-VR application and Leap Motion teleoperation	383
Summary	384
Chapter 12: Face Detection and Tracking Using ROS, OpenCV, and Dynamixel Servos	385
Technical requirements	386
Overview of the project	386
Hardware and software prerequisites	387
Installing the usb_cam ROS package	388
Creating an ROS workspace for dependencies	388
Configuring a webcam on Ubuntu 18.04	388
Interfacing the webcam with ROS	390
Configuring a Dynamixel servo using RoboPlus	392
Setting up the USB-to-Dynamixel driver on the PC	394
Interfacing Dynamixel with ROS	398
Installing the ROS dynamixel_motor packages	399
Creating face tracker ROS packages	399
The interface between ROS and OpenCV	401
Working with the face-tracking ROS package	402
Understanding the face tracker code	405
Understanding CMakeLists.txt	409
The track.yaml file	411
Launch files	411
Running the face tracker node	412
The face_tracker_control package	413
The start_dynamixel launch file	414
The pan controller launch file	415
The pan controller configuration file	415
The servo parameters configuration file	416
The face tracker controller node	416
Creating CMakeLists.txt	418
Testing the face tracker control package	419
Bringing all of the nodes together	420
Fixing the bracket and setting up the circuit	421
The final run	421
Summary	422
Other Books You May Enjoy	423

Preface

Robot Operating System (ROS) is one of the most popular robotics middleware and is used by universities and industries for robot-specific applications. Ever since its introduction, many robots have been introduced to the market and users have been able to use them with ease within their applications. One of its main draws is its open source nature. ROS does not need a user to reinvent the wheel; instead, standardizing robot operations and applications is simple.

This book is an upgrade to the previous edition and introduces you to newer ROS packages, interesting projects, and some added features. This book targets projects in the latest (at the time of writing) ROS distribution—ROS Melodic Morenia with Ubuntu Bionic version 18.04.

Here, you will understand how robots are used in industries and will learn the step-by-step procedure of building heterogeneous robot solutions. Unlike the service call and action features in ROS, you will be introduced to cooler techniques that let robots handle intricate tasks in a smart way. This knowledge should pave the way to far more intelligent and self-performing autonomous robots. Additionally, we will also introduce ROS-2, so you can learn the differences between this version and the previous ROS version and find help in choosing a specific middleware for your application.

Industries and research institutes are focusing primarily on the fields of computer vision and natural language processing. While the previous edition of this book introduced you to some simple vision applications such as object detection and face tracking, this edition will introduce you to one of the most widely used smart speaker platforms on the market, Amazon's Alexa, and how to control robots using it. In parallel, we will introduce new hardware, such as Nvidia Jetson, Asus Tinker Board, and BeagleBone Black and explore their capabilities with ROS.

While people may know how to control robots individually, one of the most common problems faced by users in the ROS community is the use of multiple robots working in synchronization, whether they are of the same type or not. This becomes complicated, as robots may follow similar topic names and may possibly lead to confusion in a sequence of operations. This book helps in highlighting the possible conflicts and suggests solutions.

This book also touches on reinforcement learning, including how it can be used with robotics and ROS. Furthermore, you will find the most interesting projects for building a self-driving car, deep learning with ROS, and building teleoperation solutions using VR headsets and Leap Motion, as they're currently trending and are being researched continuously.

Who this book is for

This book is for students, hobbyists, professionals, and individuals with a passion for learning robotics technology. Additionally, it is aimed at those individuals who are most interested in learning about and writing algorithms, motion control, and perception capabilities from scratch. This might even help a start-up build a new product or help researchers utilize what's already available and create something new and innovative. This book is also intended for those people who would like to work in the software domain or who want to have a career as a robotics software engineer.

What this book covers

Chapter 1, Getting Started with ROS, is a basic introductory chapter on ROS for beginners. This chapter will help you get an idea of the ROS software framework and its concepts.

Chapter 2, Introduction to ROS-2 and Its Capabilities, introduces you to ROS-2, the newest upgraded framework that helps us use ROS in real-time applications. This chapter is organized in a similar manner to *Chapter 1, Getting Started with ROS*, such that users are able to differentiate between both ROS versions and understand their capabilities and limitations.

Chapter 3, Building an Industrial Mobile Manipulator, is where you will learn how to build a mobile robot and a robot arm and combine them both to be used in a virtual environment and control them through ROS.

Chapter 4, Handling Complex Robot Tasks Using State Machines, introduces you to techniques in ROS that could be adapted while using robots for continuous and complicated task management.

Chapter 5, Building an Industrial Application, is where you will combine the skills acquired in *Chapter 3, Building an Industrial Mobile Manipulator* and *Chapter 4, Handling Complex Robot Tasks Using State Machines*, effectively, to create a user application. Here, we will demonstrate how to use the mobile manipulator to deliver products to houses in a neighborhood.

Chapter 6, *Multi-Robot Collaboration*, teaches you how to communicate between multiple robots of the same or different category and control them separately and together in groups.

Chapter 7, *ROS on Embedded Platforms and Their Control*, helps you understand the latest embedded controller and processor boards, such as STM32-based controllers, Tinker Board, Jetson Nano, and many more. We will also look at how to control their GPIOs via ROS and control them via voice-based commands through Alexa.

Chapter 8, *Reinforcement Learning and Robotics*, introduces you to one of the most commonly used learning techniques in robotics called reinforcement learning. In this chapter, you will understand what reinforcement learning is and the math behind it using examples. Additionally, we will discover how to incorporate this learning technique with ROS by means of simple projects.

Chapter 9, *Deep Learning Using ROS and TensorFlow*, is a project made using a trending technology in robotics. Using the TensorFlow library and ROS, we can implement interesting deep learning applications. You can implement image recognition using deep learning, and an application using SVM can be found in this chapter.

Chapter 10, *Creating a Self-Driving Car Using ROS*, is one of the more interesting projects in this book. In this chapter, we will build a simulation of a self-driving car using ROS and Gazebo.

Chapter 11, *Teleoperating Robots Using a VR Headset and Leap Motion*, shows you how to control a robot's actions using a VR headset and Leap Motion sensor. You can play around with VR, which is a trending technology these days.

Chapter 12, *Face Detection and Tracking Using ROS, OpenCV, and Dynamixel Servos*, takes you through a cool project that you can make with ROS and the OpenCV library. This project basically creates a face tracker application in which your face will be tracked in such a way that the camera will always point to your face. We will use intelligent servos such as Dynamixel to rotate the robot on its axis.

To get the most out of this book

- You should have a powerful PC running a Linux distribution, preferably Ubuntu 18.04 LTS.
- You can use a laptop or desktop with a graphics card, and RAM of at least 4 to 8 GB is preferred. This is actually for running high-end simulations in Gazebo, as well as for processing point clouds and computer vision.

- You should have the sensors, actuators, and I/O boards mentioned in the book and should be able to connect them all to your PC. You also need Git installed to clone the package files.
- If you are a Windows user, it will be good to download VirtualBox and set up Ubuntu on it. However, do note that you may have issues while you try to interface real hardware to ROS when working with VirtualBox.

Download the example code files

You can download the example code files for this book from your account at www.packt.com. If you purchased this book elsewhere, you can visit www.packtpub.com/support and register to have the files emailed directly to you.

You can download the code files by following these steps:

1. Log in or register at www.packt.com.
2. Select the **Support** tab.
3. Click on **Code Downloads**.
4. Enter the name of the book in the **Search** box and follow the onscreen instructions.

Once the file is downloaded, please make sure that you unzip or extract the folder using the latest version of:

- WinRAR/7-Zip for Windows
- Zipeg/iZip/UnRarX for Mac
- 7-Zip/PeaZip for Linux

The code bundle for the book is also hosted on GitHub at <https://github.com/PacktPublishing/ROS-Robotics-Projects-SecondEdition>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at <https://github.com/PacktPublishing/>. Check them out!

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: http://www.packtpub.com/sites/default/files/downloads/9781838649326_ColorImages.pdf.

Code in Action

Visit the following link to check out videos of the code being run:

<http://bit.ly/34p6hL0>

Conventions used

There are a number of text conventions used throughout this book.

CodeInText: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "Remove the `CMakeLists.txt` file."

A block of code is set as follows:

```
def talker_main():
    rospy.init_node('rosl_talker_node')
    pub = rospy.Publisher('/chatter', String)
    msg = String()
    i = 0
```

Any command-line input or output is written as follows:

```
$ sudo apt-get update
$ sudo rosdep init
```

Bold: Indicates a new term, an important word, or words that you see on screen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Click on **Software & Updates** and enable all of the Ubuntu repositories."



Warnings or important notes appear like this.



Tips and tricks appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at customercare@packtpub.com.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit www.packtpub.com/support/errata, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at copyright@packt.com with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit authors.packtpub.com.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions, we at Packt can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about Packt, please visit packt.com.