

Programming Robots with ROS:

1. Introduction.	3
Brief History	4
Philosophy	4
Installation	6
Summary	7
2. Preliminaries.	9
The ROS Graph	9
roscore	11
catkin, Workspaces, and ROS Packages	13
catkin	13
Workspaces	13
ROS Packages	14
roslaunch	17
Names, Namespaces, and Remapping	22
roslaunch	23
The Tab Key	25
tf: Coordinate Transforms	25
Poses, Positions, and Orientations	26
tf	27
Summary	29
3. Topics.	31
Publishing to a Topic	32
Checking That Everything Works as Expected	34
Subscribing to a Topic	36
Checking That Everything Works as Expected	37
Latched Topics	38
Defining Your Own Message Types	39
Defining a New Message	41
Using Your New Message	45
When Should You Make a New Message Type?	47
Mixing Publishers and Subscribers	47
Summary	48
4. Services.	51
Defining a Service	51
Implementing a Service	55
Checking That Everything Works as Expected	56
Other Ways of Returning Values from a Service	56
Using a Service	57
Checking That Everything Works as Expected	58
Other Ways to Call Services	58
Summary	59
5. Actions.	61
Defining an Action	62
Implementing a Basic Action Server	64
Checking That Everything Works as Expected	66
Using an Action	67

Checking That Everything Works as Expected	68
Implementing a More Sophisticated Action Server	68
Using the More Sophisticated Action	71
Checking That Everything Works as Expected	72
Summary	74
6. Robots and Simulators.	77
Subsystems	77
Actuation: Mobile Platform	77
Actuation: Manipulator Arm	80
Sensors	81
Computation	87
Complete Robots	88
PR2	88
Fetch	89
Robonaut 2	90
TurtleBot	91
Simulators	92
Stage	93
Gazebo	95
Other Simulators	96
Summary	97
7. Wander-bot.	99
Creating a Package	99
Reading Sensor Data	103
Sensing and Actuation: Wander-bot!	106
Summary	108
Part II. Moving Around Using ROS	
8. Teleop-bot.	111
Development Pattern	112
Keyboard Driver	112
Motion Generator	114
Parameter Server	119
Velocity Ramps	122
Let's Drive!	125
rviz	126
Summary	134
9. Building Maps of the World.	135
Maps in ROS	135
Recording Data with rosbag	138
Building Maps	140
Starting a Map Server and Looking at a Map	147
Summary	150
10. Navigating About the World.	151
Localizing the Robot in a Map	151
Getting a Good Initial Localization	154
What's Going on Behind the Scenes	155
Tips for Setting a Better Initial Pose	156
Using the ROS Navigation Stack	156

The ROS Navigation Stack	157
Navigating in rviz	157
Seeing What's Going On	158
Navigating in Code	161
Summary	163
11. Chess-bot.	165
Joints, Links, and Kinematic Chains	166
Joint Space	167
Inverse Kinematics	169
The Key to Success	170
Installing and Running a Simulated R2	172
Moving R2 from the Command Line	175
Moving R2 Around a Chessboard	177
Operating the Hand	179
Modeling a Chessboard	182
Playing Back a Famous Chess Game	185
Summary	188
Part III. Perception and Behavior	
12. Follow-bot.	193
Acquiring Images	193
Detecting the Line	200
Following the Line	206
Summary	208
13. On Patrol.	209
Simple Patrolling	209
State Machines	211
State Machines in ROS	212
Defining State Machines with smach	213
A Slightly More Relevant Example	216
Defining State Machines Procedurally	219
Patrolling with State Machines	221
Patrolling with State Machines	221
A Better Way to Patrol	222
Summary	224
14. Stockroom-bot.	225
Stockroom Simulation	225
Driving to Bins	238
Picking Up the Item	242
Summary	256
Part IV. Bringing Your Own Stuff into ROS	
15. Your Own Sensors and Actuators.	259
Adding Your Own Sensors	259
A (Fake) Sensor	259
Designing the ROS Wrapper	260
Design 1: Periodic Measurements over a Topic	261
Design 2: Streaming Measurements over a Topic	263
Design 3: Streaming Measurements Published at a Fixed Rate	264
Design 4: Sensor Measurements on Demand	266

Adding Your Own Actuators	267
A (Fake) Actuator	267
Designing the ROS Wrapper	268
Design 1: Continuous Actuation	270
Design 2: Infrequent, Instantaneous Actuation	271
Design 3: Infrequent, Extended Actuation	272
Summary	274
16. Your Own Mobile Robot.	275
TortoiseBot	275
ROS Message Interface	277
Hardware Driver	280
Modeling the Robot: URDF	281
Simulation in Gazebo	289
Summary	297
17. Your Own Mobile Robot: Part 2.	299
Verifying Transforms	299
Adding a Laser Sensor	304
Configuring the Navigation Stack	308
Using rviz to Localize and Command a Navigating Robot	313
Summary	317
18. Your Own Robot Arm.	319
CougarBot	319
ROS Message Interface	321
Hardware Driver	322
Modeling the Robot: URDF	323
Simulation in Gazebo	327
Verifying Transforms	335
Configuring MoveIt	339
Using rviz to Send Goals	343
Summary	345
19. Adding a Software Library.	347
Make Your Robot Talk: pyttss	348
Action Interface	349
Parameters	350
Event Loops	351
The Speech Server	351
The Speech Client	354
Checking That Everything Works as Expected	354
Summary	355
Part V. Tips and Tricks	
20. Tools.	359
The Master and Friends: roscore	359
Parameters: rosparam	361
Navigating the Filesystem: roscd	362
Starting a Node: rosrn	362
Starting Many Nodes: roslaunch	363
Testing a Many-Node System: rostest	366
Introspection: rosnod, rostopic, rosmg, rosservice, and rossrv	369

Summary	373
21. Debugging Robot Behavior.	375
Log Messages: /rosout and rqt_console	375
Generating Log Messages: /rosout	376
Logger Levels	378
Reading Log Messages: rqt_console	380
/rosout Versus /rosout_agg	382
Nodes, Topics, and Connections: rqt_graph and rosnode	383
Visualizing the Graph: rqt_graph	383
Problem: Mismatched Topic Names	385
Problem: Mismatched Topic Types and/or Checksums	386
Problem: Incorrect Network Settings	389
Sensor Fusion: rviz	391
Plotting Data: rqt_plot	392
Data Logging and Analysis: rosbag and rqt_bag	394
Logging and Playing Back Data: rosbag	394
Visualizing Bags: rqt_bag	397
Analyzing ROS Bags with Other Tools: rostopic echo -b	397
Summary	398
22. The ROS Community: Online Resources.	399
Etiquette	399
The ROS Wiki	400
ROS Answers	401
Trackers (Bugs and Feature Requests)	403
Mailing Lists and Special Interest Groups	403
Finding and Sharing Code	404
Summary	404
23. Using C++ in ROS.	405
When Should You Use C (or Some Other Language)?	406
Building C++ with catkin	406
package.xml	407
CMakeLists.txt	407
catkin_make	408
Translating from Python to C++ (and Back Again)	408
A Simple Node	409
Topics	410
Services	412
Summary	414
Index.	417

A Systematic Approach to Learning Robot Programming with ROS:

ABB IRB120 robot kinematics, 92	action client sendGoal, 82	action message header files, 67
Ackermann steering, 330	action client threading, 83	action message instantiation, 70
acquire block client, 474	action client waitForServer(), 82	action message name mangling, 70
action client, 65, 72	action goal, result, feedback, 66	action server, 65
action client blocking, 82	action message, 65	action server callback, 70
action client callback functions, 81	action message dependency, 74	action server feedback, 77
	action message generation, 67	

- action server goal cancellation, 79
- action server reconnection, 77
- action server start-up, 71
- action server topics, 75
- action server, multiple clients, 83
- action setSucceeded(), 71
- actionlib, 73
- Adaptive Monte-Carlo Localization, 325
- add executable, 14
- AMCL, 325, 353
- AMCL, initialization, 327
- apply joint effort, 124
- approach pose, 444
- arm motion planning, 401
- base link, 105
- base local planner
 - params.yaml, 354
- baseLocalPlanner, 361
- Baxter enable robot, 414
- Baxter robot simulator, 413
- Baxter, approximate inverse kinematics, 416, 433
- Baxter, baxter model
 - variations, 472
- Baxter, baxter examples, 414
- Baxter, baxter playback, 428
- Baxter, baxter tools, 414
- Baxter, Cartesian moves, 434
- Baxter, Cartesian planner, 434
- Baxter, example generic cartesian move client, 438
- Baxter, gripper control, 418
- Baxter, head pan control, 421
- Baxter, joint commands, 422
- Baxter, joint names, 417
- Baxter, joint states, 416
- Baxter, joint trajectory controller, 425
- Baxter, joints and topics, 415
- Baxter, kinematics, 432
- Baxter, on mobile base, 481
- Baxter, reachable poses, 437
- Baxter, right-arm Cartesian move action server, 437
- Baxter, with Kinect sensor, 482
- baxter core msgs, 422
- baxter fk ik, 433
- baxter trajectory streamer, 423
- callback, 21, 50
- callback, action server, 70

- camera calibration, 223
- camera calibration template, 227
- camera calibration YAML file, 230
- camera frame, 224
- camera image rectification, 207
- camera matrix, 236
- camera model, 207
- camera model, distortion, 225
- camera model, focal length, 224
- camera model, intrinsic parameters, 223
- camera model, optical axis, 223
- camera model, optical center, 224
- camera model, principal plane, 224
- camera model, projection center, 223
- camera parameters, 207
- camera projection matrix, 228
- camera topic, image raw, 207
- camera topics, 227
- camera calibration package, 225
- camera depth optical frame, 267
- camera info directory, 230
- cameracalibrator.py node, 234
- cameras, simulation in Gazebo, 205
- Cartesian motion action server, 408
- Cartesian motion planning, 402
- Cartesian move client node, 411
- catkin create pkg, 9
- catkin make, 14
- catkin simple, 8, 24
- check urdf, 107
- checkerboard model, 228
- child frame, 156
- child frame id, 163
- chrony, 168
- class constructor, 56
- class header, 54
- class implementation, 56
- classes in ROS, 54
- client, 52
- clients, 47
- closed-chain mechanism, 105

- CMakeLists, 8, 22
- CMakeLists.txt, 25
- CMakeLists.txt and OpenCV, 238
- CMakeLists.txt with PCL, 265
- CMakeLists.txt, library linking, 63
- cmd vel, 97, 141
- cmd vel open-loop control, 299
- collision model, 103, 112
- command bundler, 474
- constructor initializer list, 58
- control gain YAML file, 373
- control gains, tuning, 373
- control parameters, 130
- control
 - msgs/FollowJointTrajectoryAction, 426
- controller namespace, 130
- controller spawner, 131
- coordinate frame, 153
- coordinate transforms, 153
- coordinator acquire block client, 478
- coordinator command bundler, 483
- coordinator dropoff block client, 480
- coordinator, acquire block client, 483
- coordinator, dropoff block client, 483
- coordinator, fetch and stack client, 484
- costmap, 353
- costmap, configuration files, 354
- costmap, global, 354
- costmap, local, 354
- costmap common
 - params.yaml, 354
- cs add executable, 26
- cs add library, 63
- cs create pkg, 9, 24
- custom messages, 38
- Denavit-Hartenberg, 390
- Denavit-Hartenberg parameters, 106
- depart pose, 444
- dept cameras, 258
- depth camera simulation, 209
- depth from stereo cameras, 252

- depth imaging, 247
- desired-state generator, 292
- DH parameters, 106
- differential-drive controller, 133
- differential-drive kinematics, 311
- differential-drive steering algorithms, 330
- disparity image, 255
- disparity, stereo cameras, 253
- displacements and efforts, 382
- dynamic model, 109
- dynamic programming, 403
- E-stop reset behavior, 305
- effort, 129
- effort control, 128
- EffortJointInterface, 129, 372
- Eigen and CMakeLists.txt, 168
- Eigen initializations, 169
- Eigen library, 168
- Eigen, cross product, 170
- Eigen, dot product, 170
- Eigen, plane fitting, 171
- Eigen::Affine3d, 172
- Eigen::Solver, 172
- Eigen::Matrix3d, 169
- Eigen::MatrixXd, 169
- Eigen::Vector3d, 168
- empty world, 114
- epipolar frame, 236
- estop service, 303
- example move base client, 483
- extrinsic calibration, 470
- extrinsic camera calibration, 281, 469
- feedforward graph, 404
- find package in roslaunch, 116
- fixed joint, 108
- forward kinematics, 389, 390
- frame id, 38, 163
- Gazebo, 95, 114
- Gazebo camera plug-in, 206
- Gazebo camera simulation, 205
- Gazebo Kinect simulation, 210
- Gazebo LIDAR plug-in, 201
- Gazebo material colors, 144
- Gazebo messages, 120
- Gazebo model friction, 144
- Gazebo numerical iterations, 144
- Gazebo plug-in, 122
- Gazebo sensor noise, 206

- Gazebo services, 121
- Gazebo set model state, 117
- Gazebo stereo-camera plug-in, 233
- Gazebo time step, 144
- Gazebo, collisions, 127
- Gazebo, control plug-in, 128
- Gazebo, force/torque sensor plug-in, 378
- Gazebo, getting model state, 308
- Gazebo, IMU plug-in, 320
- Gazebo, inserting models, 115
- Gazebo, menus, 115
- Gazebo, plug-ins, 371
- Gazebo, real-time factor, 114
- Gazebo, reset model poses, 456
- Gazebo, spawn model, 115
- Gazebo, temporary joint, 447
- Gazebo, view contacts, 144
- Gazebo, view joints, 141
- Gazebo_ros, 202
- Generic_gripper_services, 447
- Generic_gripper_frame, 410
- Generic_gripper_services, 421
- Geometry_msgs, 97
- Geometry_msgs/PoseWithCovarianceStamped, 327
- Geometry_msgs/WrenchStamped, 378
- geometry_msgs_PoseStamped, 163
- get_joint_properties, 124
- getParam(), 87
- global costmap, 354
- global planner, 354
- global costmap params.yaml, 354
- gmapping, 347
- goal cancellation, 79
- GPS, 320
- GPU and LIDAR simulation, 202
- grasp pose, 444
- grasp strategy, 445
- gravity in Gazebo, 115
- gripper abstraction, 442, 443
- gripper frame transforms, 446
- gripper ID, 444
- gzclient, 114
- gzserver, 114
- heading error periodicity, 331
- holonomic, 356

- homogeneous transformation matrix, 154
- image throttling, 243
- image transport, 238, 254
- image view, 208
- imager rate, 243
- IMU, 317
- inertia tensor, 110
- inertial frame, 110
- Inertial Measurement Unit, 317
- inertial model, 103, 109
- inserting models in Gazebo, 115
- integral-error gain, 131
- integrated perception and manipulation, 472
- interactive marker handles, 198
- interactive marker pose query, 197
- interactive markers, 191
- InteractiveMarkerServer, 197
- intrinsic camera calibration, 225
- inverse kinematics, 389, 394
- inverse kinematics, ABB IRB120, 396
- inverse kinematics, analytic, 390
- inverse kinematics, numerical, 390
- inverse kinematics, rrbot, 394
- Jacobian, 399
- joint angle, 107
- joint control topics, 132
- joint controller, 122
- joint frame, 107
- joint home angle, 107
- joint names, 383
- joint state, 125
- joint transmission, 128
- joint, continuous, 129
- joint, fixed, 108, 146
- joint, limits, 129
- joint, prismatic, 129
- joint, revolute, 107, 129
- joint-space interpolation, 384
- joint-space planner, cost function, 405
- joint-space planner, cost to go, 405
- joint-space planning, 401, 403
- joint-state publisher, 159
- joint states, 179, 181
- joint states topic, 157

- Joint Trajectory Action Server, 385
- JointPositionController, 373
- JointVelocityController, 376
- Kalman filter, 325
- KDL, Kinematics and Dynamics Library, 389
- Kinect camera, 209, 258
- Kinect model, 211
- Kinect model example, 258
- kinect depth frame, 211
- kinematic model, 103
- kinematics, 389
- latched messages, 326
- launchfile, including files, 147
- libgazebo ros camera.so
- Gazebo plug-in, 206
- libgazebo ros control plug-in, 128
- libgazebo ros diff drive.so, 139
- libgazebo ros ft sensor.so
- Gazebo plug-in, 378
- libgazebo ros gpu laser.so
- Gazebo plug-in, 201
- libgazebo ros imu.so IMU plug-in, 320
- libgazebo ros joint state publisher.so, 157
- libgazebo ros multicamera.so
- Gazebo plug-in, 233
- libgazebo ros openni kinect.so
- Gazebo plug-in, 210
- libraries in ROS, 60
- library dependency, 64
- library linking, same package, 64
- library, naming, 63
- libsticky fingers.so, Gazebo plug-in, 447
- LIDAR, 96, 199
- LIDAR scan topic, 202
- LIDAR simulation, 201
- LIDAR, tilting, 247
- link frame, 107
- local costmap, 354
- local planner, 354
- local costmap params.yaml, 354
- localization, heading estimation, 322
- localization, Kalman filtering, 325
- lookupTransform(), 162, 167

- low-level joint control, 371
- manip task action service, 476
- ManipTask.action, 476
- map frame, 319
- map topic, 326
- map saver, 349
- map server, 325
- markers in rviz, 182
- message, 6
- message Header, 38
- message header file, 40
- message, variable length, 42
- message generation, 39
- message runtime, 39
- messages, 38
- messages, latched, 326
- minimal controller, 30
- minimal joint controller, 122
- minimal publisher, 11
- minimal robot description, 105
- minimal simulator, 30
- minimal nodes, 9
- mobile manipulation, 481
- mobile robot modeling, 133
- mobile robot state estimation, 308
- mobile robot steering, phase space, 333
- mobile robot, kinematic model, 331
- mobile robot, localization with noisy GPS, 320
- mobile robot, non-linear steering algorithm, 333
- mobile robot, non-linear steering dynamics, 332
- mobile robots, 287
- mobile robots, desired-state generator, 292
- mobile robots, heading, 290
- mobile robots, nav msgs/Path, 291
- mobile robots, paths and trajectories, 290
- mobile robots, traj builder, 294
- mobile-robot navigation, desired state generation, 289
- model, center of mass, 110
- model, collision, 112
- model, dynamic, 109
- model, inertial, 109
- model, inertial frame, 110
- modeling, URDF, 103

- move base, 353
- move base client, 358
- move base
- msgs/MoveBaseAction, 360
- move calibration checkerboard node, 229
- msg, 38
- namespace, robot, 132
- Natural Admittance Control, NAC, 379
- nav msgs, 102
- nav_msgs/Path, 291
- nav_msgs::Odom, 295
- navigation stack, 347
- navstack, observation sources, 355
- node, 5, 8
- node handle, 12
- node name, 15
- object finder, 474
- object grabber action client, 452
- object grabber action server, 449
- object grabber action service, 474
- object grabber package, 441
- object grabber, action client, 441
- object grabber, action server, 441
- object ID, 444
- object manipulation query service, 443
- Object Recognition Kitchen, 284
- object finder package, 280
- occupancy map, 349
- ODE, stifi contact, 143
- odom frame transformation, 319
- odom publisher node, 313
- odom to map transform, 327
- odom topic, 102
- Odometry, 102, 291
- Odometry and LIDAR, 325
- odometry drift, 317
- odometry frame, 319
- Open Dynamics Engine, 103
- open kinematic chain, 389
- open-chain mechanism, 105
- open loop controller, 306
- open loop nav service, 483

- open loop yaw service, 483
- OpenCV, 221, 238
- OpenCV and CMakeLists.txt, 238
- OpenCV, Canny edge filter, 244
- OpenCV, cv bridge, 241
- OpenCV, edge finding, 243
- OpenCV, finding colored pixels, 238
- OpenCV, tolImageMsg(), 242
- optical center, 224
- optimal path planning, 406
- orientation matrix, 154
- package, 8, 9
- package dependency, 64
- package.xml, 10, 25, 74
- parameter server, 84
- parameter server, accessing from source code, 87
- parameter server, getParam(), 87
- parameter server, robot description, 88
- parameter server, URDF file, 88
- parameter server, YAML configuration files, 85
- parent frame, 156
- path planning, 347
- path-following error coordinates, 331
- PCD file display, 267
- PCL, 221
- PCL and rviz point selection, 275
- PCL centroid, 257
- PCL in CMakeLists.txt, 265
- PCL PCD files, 265
- PCL pcl::PointXYZ, 263
- PCL pcl::PointXYZRGB, 263
- PCL plane normal, 276
- PCL, box filter, 278
- PCL, down-sampling, 273
- PCL, object finder, 280
- PCL, savePCDFile, 265
- PCL, savePCDFileASCII, 265
- PCL, table finder, 282
- PCL, transformPointCloud(), 278
- pcl::computePointNormal, 276
- pcl::copyPointCloud(), 279
- pcl::PassThrough, 278
- pcl::PassThrough filter methods, 276

- pcl::VoxelGrid, 273
- pcl_conversions, 265
- pcl_ros, 265
- pcl_utils, 257
- pcl_utils snapshot, 269
- perception-based manipulation, 469
- perceptual processing, 221
- physics engine, 103
- PID controller, 130
- PID gains, 85
- playfile, 431
- playfile service, 431
- plugin, 361
- point cloud, 209
- Point Cloud Library, 221, 261
- Point Cloud Processing, 261
- point-cloud point selection in rviz, 216
- PointCloud object, 263
- PointCloud, ordered, 268
- PointCloud, unordered, 263
- PointCloud2, 213
- polyline path, 292
- PoseArray, 326
- prismatic joint, 129
- projection center, 223
- projection matrix, 228
- projective transformation, 223
- pub des state, 301
- publish, 13
- publish selected points, 214, 252
- publisher, 11
- PublishSelectedPoints tool, 215
- publishWheelJointState, 156
- publishWheelTF, 156
- push back, 43
- quaternion, 156, 157
- quaternion, conversion to heading, 290
- Rate, 19
- rectification, stereo cameras, 234
- rectified images, 207
- redundant kinematics, 389
- remapping, 142
- request, service, 47
- response, service, 47
- revolute joint, 107, 129
- robot abstraction, 409
- robot arm abstraction, 442
- robot arm kinematics, 389

- robot arm motion planning and control, 369
- robot footprint, 355
- robot joint, 105
- robot link, 105
- robot modeling, 103
- robot namespace, 132
- robot state, 291
- robot state estimation, with GPS and IMU, 319
- robot description, 88, 120
- robot state publisher, 159, 164, 179
- ROS environment, 8
- ROS workspace, 8
- ros workspace, 39
- ROS-Industrial, 385
- ros::init, 12
- ros::ok, 13
- ros::Publisher, 10
- ros::Rate, 19
- ros::ServiceClient, 53
- ros::ServiceServer, 50
- ros::spin, 22
- ros::Subscriber, 10, 21
- ros::Time, 41
- ros::Time(0), 163
- ros controllers, 128
- ROS_INFO, 21
- ROS_INFO_STREAM, 21
- ROS_INFO_STREAM(), 170
- ROS_WARN(), 88
- ROS_WORKSPACE, 116
- roscpp, 28
- roscd, 9
- roscore, 15, 27
- roscpp, 9
- roslaunch, 26
- roslaunch screen output, 132
- roslaunch, find package, 116
- roslaunch, parameter loading, 86
- roslaunch, spawn model, 116
- rosmmsg show, 13, 38
- rosmode, 8
- rosmode list, 18, 27
- rosout, 18
- rospack find, 116
- rosparam delete, 88
- rosparam load in roslaunch, 86
- rosparam set, get, load, list, 85
- roslaunch, 8, 15
- rosservice call, 51, 59

- rostopic, 8, 16
- rostopic echo, 19
- rostopic hz, 19
- rostopic pub, 60
- rqt console, 27
- rqt graph, 8, 23
- rqt gui, 373
- rqt gui, dynamic reconfigure, 374
- rqt plot, 32, 125
- rqt reconfigure, 131
- rqt reconfigure, stereo imaging, 255
- rrbot, 390
- rviz, 177
- rviz and rosbag, 179
- rviz as operator interface, 217
- rviz camera display item, 208
- rviz configuration file, 204
- rviz display PointCloud2, 213
- rviz fixed frame, 181
- rviz interactive markers, 191
- rviz InteractiveMarkers display, 198
- rviz LaserScan display, 203
- rviz marker display item, 185
- rviz markers, 182
- rviz particlecloud, 326
- rviz publish selected points tool, 214
- rviz scan topic, decay time, 251
- rviz selected points publisher, 214
- rviz sensor visualization, 199
- rviz triad display, 185
- rviz, 2D Nav Goal, 354
- rviz, 2D pose estimate, 327
- rviz, axes display, 180
- rviz, displaying Kinect data, 270
- rviz, installing tools, 214
- rviz, map display, 349
- rviz, PointCloud2 display, 257
- rviz, robot model, 178
- rviz, selectable, 215
- rviz, starting from launchfile, 204
- rviz, stereo imaging display, 257
- scan LIDAR topic, 202
- SDF, 104
- sendGoal, 74
- sensor noise simulation, 206
- sensor visualization, 199

- sensor msgs/CameraInfo, 207, 227
- sensor msgs/Image, 207
- sensor msgs/LaserScan, 200
- sensor msgs::JointState, 386
- service call, 53
- service message, 47
- service message header file, 49, 51
- service request, 53
- service response, 53
- services, 47
- SetModelState, 117
- setting model state from program, 118
- setting model state in Gazebo, 118
- Simple Two-Dimensional Robot, 95
- SimpleActionClient, 73
- SimpleActionServer, 70
- simulation, 93
- simulation description format, 104
- simulation, dynamic model, 103
- singular pose, 399
- skid steering, 330
- SLAM, Simultaneous Localization and Mapping, 347
- slam gmapping, 347
- spawn model from file, 119
- spawn model from launch file, 120
- spawn model from parameter server, 120
- spawn model from roslaunch, 116
- spawn model in Gazebo, 115
- spherical wrist, 389
- spin, 22
- spinOnce, 31
- srv file, 47
- state estimation from odom, 313
- static map, 356
- static transform publisher, 212, 410
- std msgs, 9, 38
- std srvs, 59
- STDR, 95
- steering, 330
- steering algorithms, 330

- steering and amcl, 340
- steering linear algorithm, 332
- stereo camera calibration, 231
- stereo camera calibration file, 235
- stereo camera calibration, fix-aspect-ratio, 234
- stereo camera calibration, fix-principal point, 234
- stereo camera calibration, zero-tangent-dist, 234
- stereo camera frame
- synchronization, 237
- stereo camera rectification, 234
- stereo cameras, baseline, 255
- stereo cameras, correspondence problem, 254
- stereo cameras, disparity, 253
- stereo cameras, pointCloud2, 256
- stereo cameras, triangulation, 252
- stereo image proc, 254
- stereo image proc correlation window size, 255
- stiff contact, 143
- subscriber, 20
- task-space programming, 409
- teach and playback, 426
- teleop twist keyboard, 143, 327
- tf and geometry msgs type conversions, 167
- tf conversion to Eigen Affine, 172
- tf package, 155
- tf tf echo, 166
- tf topic, 155
- tf, try/catch, 164
- tf2 msgs, 155
- tf::StampedTransform, 162
- tf::Transform, multiplication, 163
- tf::TransformListener, 162
- tf_echo, 393
- tfListener lookupTransform, 438
- time stamp, 38
- time synchronization, 168
- tool transform, 437
- toolange, 391
- topic, 13, 21
- topic remapping, 142, 212
- trajectory messages, 382

- trajectory planning, 382
- trajectory
 - msgs/JointTrajectory, 382
 - trajectory
 - msgs/JointTrajectoryPoint, 383
- transform conversions, 173
- transform listener, 161
- transformPose(), 162, 163
- TransformStamped, 155
- transmission, 129
- trapezoidal velocity profile, 293
- triangular velocity profile, 298

- Twist, 97
- Twist command, 143
- Unified Robot Description
 - Format, 105
 - URDF, 103, 105
 - URDF file on parameter server, 88
 - URDF, combining models, 145
 - urdf to graphviz, 140
 - vehicle track, 312
 - velocity controller, 376
 - virtual vacuum gripper, 447

- visual model, 103, 108
- visualization, 93
- visualization msgs, 184
- waitForResult(), 74
- waitForServer(), 74, 82
- wheel speed from vel cmd, 318
- world frame, 319
- world model, 114
- xacro, 133, 134
- xacro macro, 135
- XformUtils, 173
- YAML, 85, 118

Artificial Intelligence for Robotics:

Chapter 1: Foundation for Advanced Robotics and AI 6

- Technical requirements 7
- The basic principle of robotics and AI 7
- What is AI (and what is it not)? 8
- There is nothing new under the sun 10
- The example problem – clean up this room! 11
- What you will learn 13
- Artificial intelligence and advanced robotics techniques 14
- Introducing the robot and our development environment 16
- Software components (ROS, Python, and Linux) 18
- Robot control systems and a decision-making framework 20
- Soft real-time control 20
- Control loops 21
- The robot control system – a control loop with soft real-time control 32
- Reading serial ports in a real-time manner 35
- Summary 39
- Questions 39
- Further reading 40

Chapter 2: Setting Up Your Robot 41

- Technical requirements 41
- What is a robot? 42
- Robot anatomy – what are robots made of? 42
- Subsumption architecture 46
- Software setup 49
- Laptop preparation 50
- Installing Python 50
- Installing ROS on the laptop 51
- Setup of Raspberry Pi 3 54
- VNC 55
- Setting up catkin workspaces 56
- Hardware 57
- Beginning at the beginning – knolling 57
- Assembling the tracks 58
- Mounting the tracks 60
- Arm base assembly (turntable) 62

Arm assembly 64

Wiring 67

Chapter 3: A Concept for a Practical Robot Design Process 70

A systems engineering-based approach to robotics 70

Our task – cleaning up the playroom 71

Use cases 72

The problem – part 1 72

Who – the robot 74

What – pick up toys and put them in the toy box 74

What – pick up and put away in the toy box the items that were not previously in the room 75

When – after the grandchildren have visited and they have left, continue to pick up toys until there are none left 75

When – when I (the user) tell you to, and don't stop until there are no more toys to be found 75

Where – the game room upstairs 76

The problem – part 2 77

Who – the robot, the user (granddad), and the grandchildren 77

What – receive commands and verbally interact (hold a conversation) with children, which must include knock-knock jokes 78

When – as requested by the robot controller, then when the child speaks to the robot 79

Where – in the game room, within about six feet of the robot 79

What is our robot to do? 79

Storyboards 83

Storyboard – put away the toys 83

Project goals 95

Decomposing hardware needs 95

Breaking down software needs 96

Writing a specification 99

Chapter 4: Object Recognition Using Neural Networks and Supervised Learning 103

Technical requirements 104

The image recognition process 104

The image recognition training and deployment process – step by step 106

Image processing 107

Convolution 108

Artificial neurons 110

The convolution neural network process 112

Build the toy/not toy detector 122

Using the neural network 131

Chapter 5: Picking up the Toys 136

Technical requirements 137

Task analysis 138

Setting up the solution 139

How do we pick actions? 141

Summary of robot arm learning process 142

- Teaching the robot arm 143
 - Version one – action state reinforcement learning 143
 - Adaptive learning rate 148
 - Q-learning implementation 149
 - Version 2 – indexed states and actions 154
 - Genetic algorithms 158
 - Other robot arm machine-learning approaches 166
 - Google’s SAC-X 166
 - Amazon Robotics Challenge 167
 - Summary 167
 - Questions 168
 - Further reading 168

Chapter 6: Teaching a Robot to Listen 169

- Technical requirements 170
- Robot speech recognition 170
- What are we doing? 170
- Speech to text 171
- Intent 175
- Mycroft 176
- Hardware 176
- Mycroft software 179
- Skills 181
- Dialogs 181
- Telling jokes – knock, knock 185
- Receiving jokes – who’s there? 189
- Summary 193
- Questions 193
- Further reading 194

Chapter 7: Avoiding the Stairs 195

- Technical requirements 196
- Task analysis 196
- What is SLAM? 198
- Alternatives for navigation 200
- Neural networks 207
- Processing the image 209
- Training the neural network for navigation 212
- Convolutional neural network robot control implementation 217
- Summary 221
- Questions 222
- Further reading 222

Chapter 8: Putting Things Away 223

- Technical requirements 224
- Task analysis 225
- Decision trees 225
- What do we mean by pruning? 227
- Self-classifying decision trees and AI tools 230
- Entropy 238
- One hot encoding 239

Random forests 244
Grid searching and A* (A-Star) 245
The A* algorithm 251
D* (D-Star or Dynamic A*) 254
GPS path finding does not use a map! 255
Summary 257
Questions 258
Further reading 258

Chapter 9: Giving the Robot an Artificial Personality 259

Technical requirements 259
What is an artificial personality? 260
The Turing test 262
The art and science of simulation 263
An emotion state machine 267
Playing the emotion game 270
Creating a model of human behavior 273
Integrating artificial personality into our robot 273
Personality construction – building blocks 274
Context 278
Under construction 279
The robot emotion engine 284
The human emotion model 286
Human information storage 287
Context memory 288
Questions 289
Further reading 290

Chapter 10: Conclusions and Reflections 291

Conclusions about our journey 294
Careers in robotics 294
Issues in AI – real and not real 296
Some very brief words about robots and AI phobia 298
Understanding risk in AI 301
Final words 303
Summary 303

Deep Learning with Keras:

1. Neural Networks Foundations Perceptron

The first example of Keras code
Multilayer perceptron — the first example of a network
Problems in training the perceptron and a solution
Activation function — sigmoid
Activation function — ReLU
Activation functions
A real example — recognizing handwritten digits
One-hot encoding — OHE
Defining a simple neural net in Keras
Running a simple Keras net and establishing a baseline
Improving the simple net in Keras with hidden layers

Further improving the simple net in Keras with dropout
Testing different optimizers in Keras
Increasing the number of epochs
Controlling the optimizer learning rate
Increasing the number of internal hidden neurons
Increasing the size of batch computation
Summarizing the experiments run for recognizing handwritten charts
Adopting regularization for avoiding overfitting
Hyperparameters tuning
Predicting output
A practical overview of backpropagation
Towards a deep learning approach
Summary

2. Keras Installation and API

Installing Keras
Step 1 — install some useful dependencies
Step 2 — install Theano
Step 3 — install TensorFlow
Step 4 — install Keras
Step 5 — testing Theano, TensorFlow, and Keras
Configuring Keras
Installing Keras on Docker
Installing Keras on Google Cloud ML
Installing Keras on Amazon AWS
Installing Keras on Microsoft Azure
Keras API
Getting started with Keras architecture
What is a tensor?
Composing models in Keras
Sequential composition
Functional composition
An overview of predefined neural network layers
Regular dense
Recurrent neural networks — simple, LSTM, and GRU
Convolutional and pooling layers
Regularization
Batch normalization
An overview of predefined activation functions
An overview of losses functions
An overview of metrics
An overview of optimizers
Some useful operations
Saving and loading the weights and the architecture of a model
Callbacks for customizing the training process
Checkpointing
Using TensorBoard and Keras
Using Quiver and Keras
Summary

3. Deep Learning with ConvNets

Deep convolutional neural network — DCNN

Local receptive fields

Shared weights and bias

Pooling layers

Max-pooling

Average pooling

ConvNets summary

An example of DCNN — LeNet

LeNet code in Keras

Understanding the power of deep learning

Recognizing CIFAR-10 images with deep learning

Improving the CIFAR-10 performance with deeper a network

Improving the CIFAR-10 performance with data augmentation

Predicting with CIFAR-10

Very deep convolutional networks for large-scale image recognition

Recognizing cats with a VGG-16 net

Utilizing Keras built-in VGG-16 net module

Recycling pre-built deep learning models for extracting features

Very deep inception-v3 net used for transfer learning

Summary

4. Generative Adversarial Networks and WaveNet

What is a GAN?

Some GAN applications

Deep convolutional generative adversarial networks

Keras adversarial GANs for forging MNIST

Keras adversarial GANs for forging CIFAR

WaveNet — a generative model for learning how to produce audio

Summary

5. Word Embeddings

Distributed representations

word2vec

The skip-gram word2vec model

The CBOW word2vec model

Extracting word2vec embeddings from the model

Using third-party implementations of word2vec

Exploring GloVe

Using pre-trained embeddings

Learn embeddings from scratch

Fine-tuning learned embeddings from word2vec

Fine-tune learned embeddings from GloVe

Look up embeddings

Summary

6. Recurrent Neural Network — RNN

SimpleRNN cells

SimpleRNN with Keras — generating text

RNN topologies

Vanishing and exploding gradients

Long short term memory — LSTM

LSTM with Keras — sentiment analysis

Gated recurrent unit — GRU

GRU with Keras — POS tagging

Bidirectional RNNs

Stateful RNNs

Stateful LSTM with Keras — predicting electricity consumption

Other RNN variants

Summary

7. Additional Deep Learning Models

Keras functional API

Regression networks

Keras regression example — predicting benzene levels in the air

Unsupervised learning — autoencoders

Keras autoencoder example — sentence vectors

Composing deep networks

Keras example — memory network for question answering

Customizing Keras

Keras example — using the lambda layer

Keras example — building a custom normalization layer

Generative models

Keras example — deep dreaming

Keras example — style transfer

Summary

8. AI Game Playing

Reinforcement learning

Maximizing future rewards

Q-learning

The deep Q-network as a Q-function

Balancing exploration with exploitation

Experience replay, or the value of experience

Example - Keras deep Q-network for catch

The road ahead

Summary

9. Conclusion

Keras 2.0 — what is new

Installing Keras 2.0

API changes