Welcome to:

Robot Operating System (ROS)

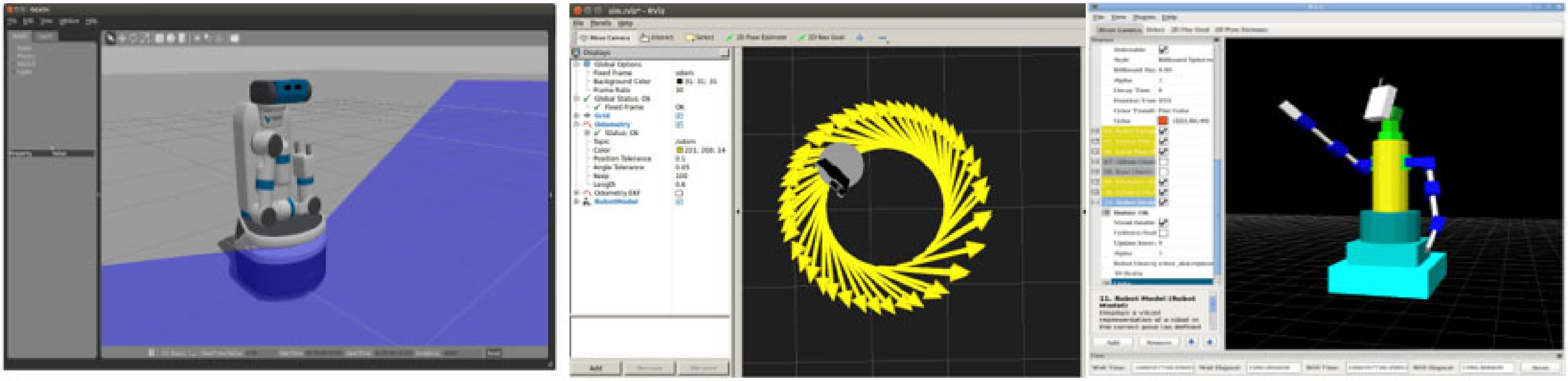
Unit objectives

After completing this unit, you should be able to:

- Understand the operating system concepts for robotics
- Gain knowledge on debugging and visualization
- Understand the concept of 3D modeling and simulation
- Gain an insight into computer vision applications for robotics

Real and simulated robots

(a) Real Robots



(b) Simulated robots

Figure: Real and Simulated Robots

Source: https://robots.ieee.org/learn/types-of-robots/ https://docs.fetchrobotics.com/gazebo.html, https://www.pirobot.org/blog/0014/

Robot Operating System (ROS)

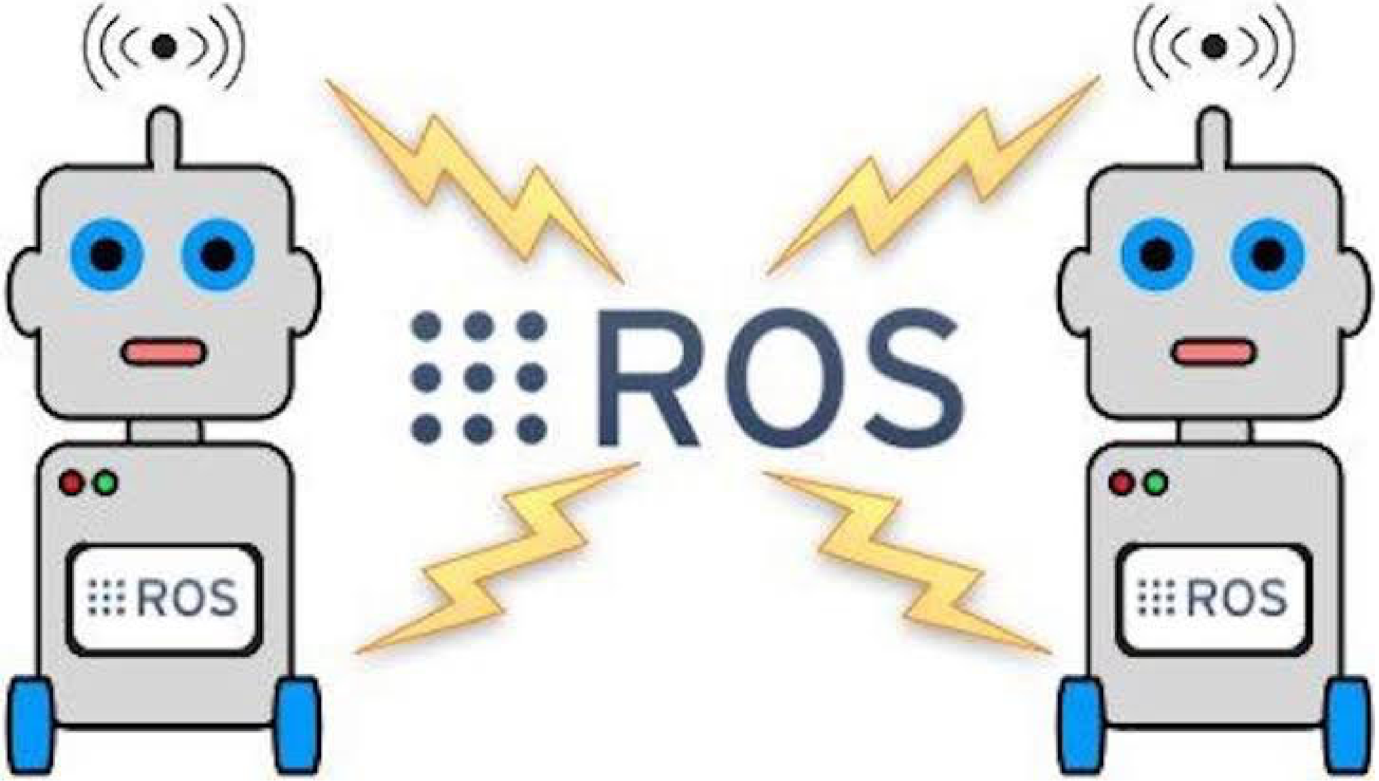


Figure: Robot Operating System (ROS)
Source: https://roboticsandautomationnews.com/2019/05/16/the-rise-of-the-robot-operating-system/22485/

ROS basics and architecture

- ROS Architecture: is divided into three levels of concept or sections. These are:
  - File system level.
  - Computation graph level.
  - Community level.

The File system level

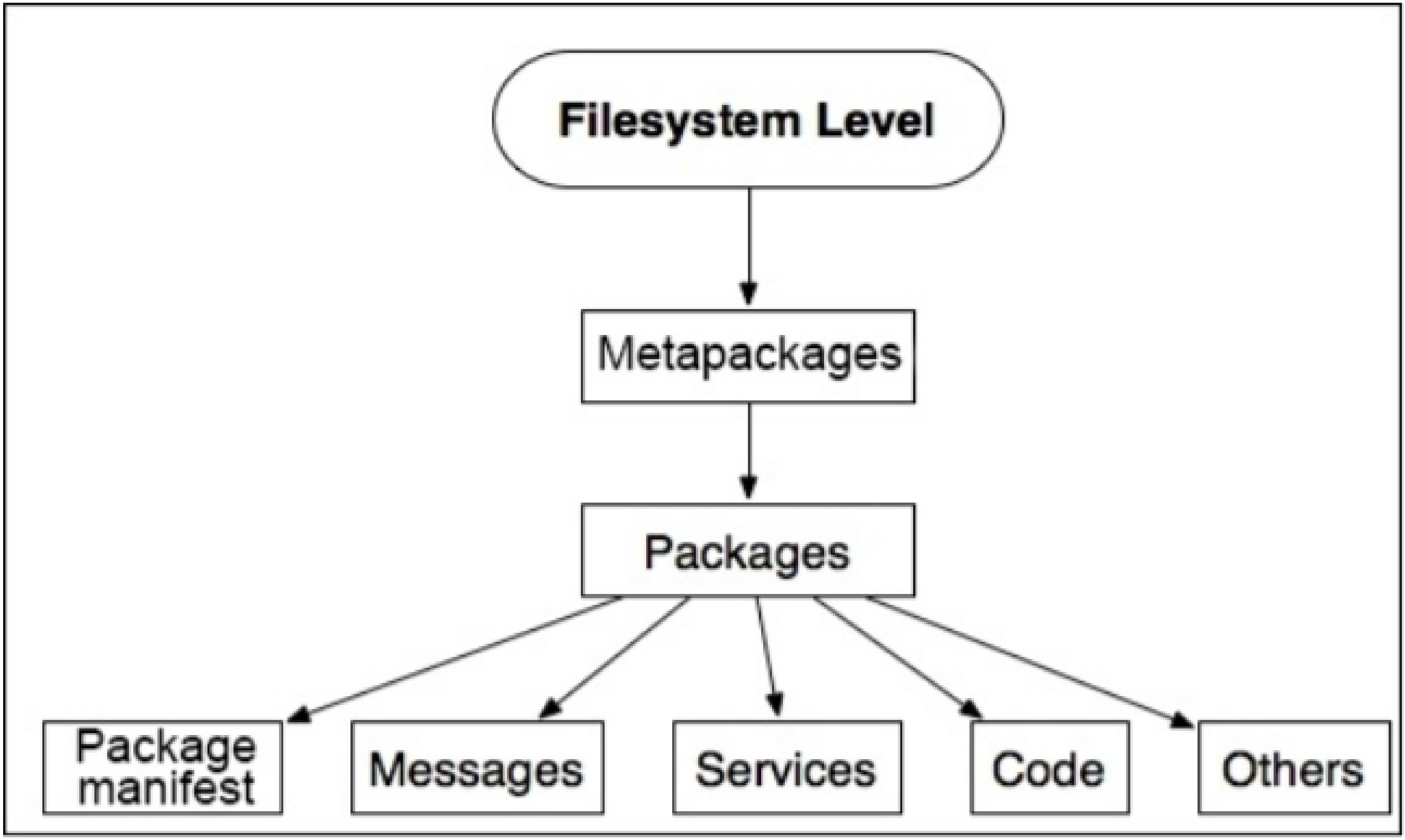
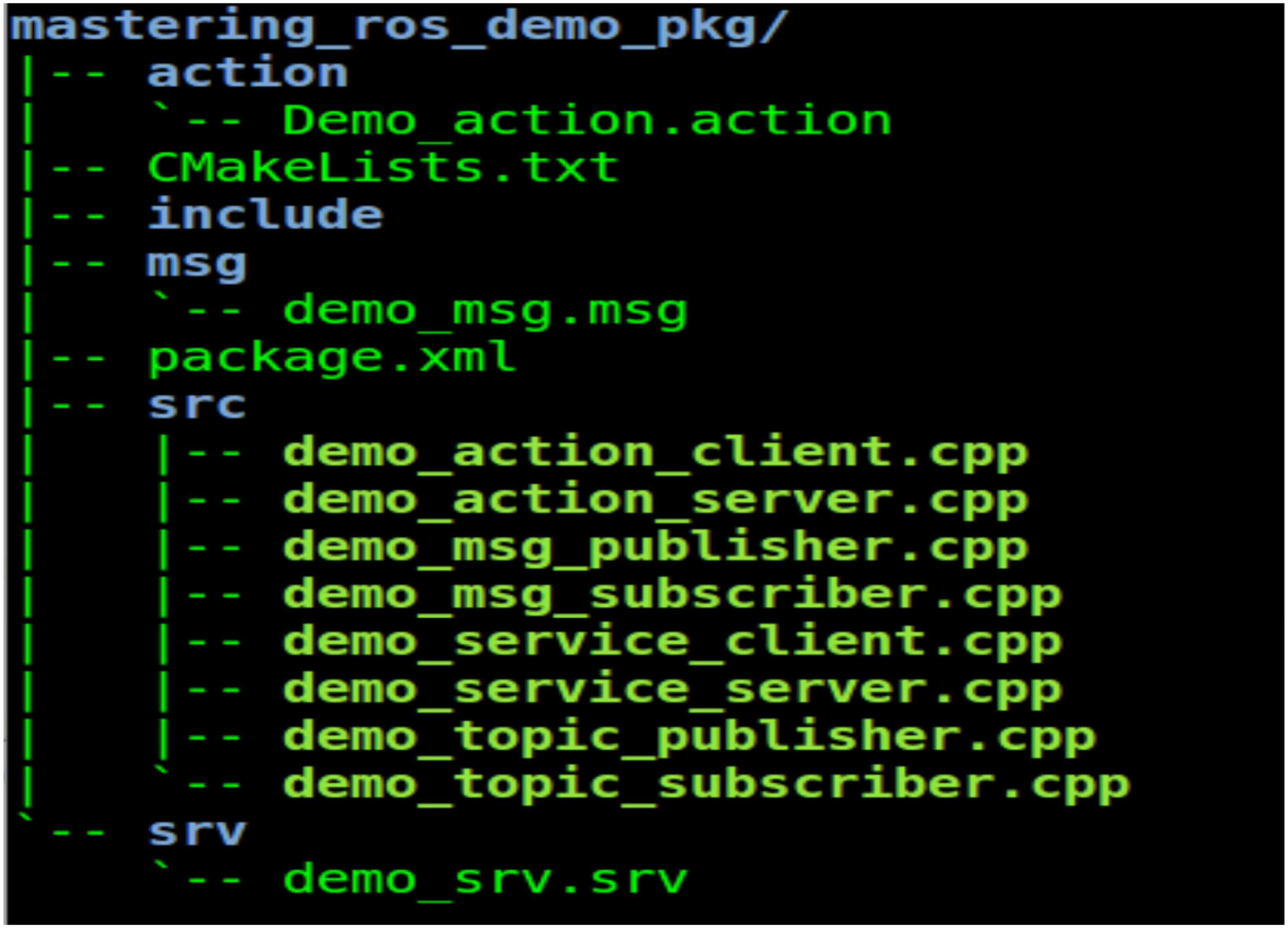


Figure: ROS File System Level

Source: (Ref: http://marte.aslab.upm.es/redmine/files/dmsf/p_drone-testbed/170324141025_280_Mahtani_-_Effective_Robotics_Programming_with_ROS.pdf)

Files and folders in a sample package of ROS



```
mastering_ros_demo_pkg/
|-- action
|   `-- Demo_action.action
|-- CMakeLists.txt
|-- include
|-- msg
|   `-- demo_msg.msg
|-- package.xml
|-- src
|   |-- demo_action_client.cpp
|   |-- demo_action_server.cpp
|   |-- demo_msg_publisher.cpp
|   |-- demo_msg_subscriber.cpp
|   |-- demo_service_client.cpp
|   |-- demo_service_server.cpp
|   |-- demo_topic_publisher.cpp
|   `-- demo_topic_subscriber.cpp
`-- srv
    `-- demo_srv.srv
```

Figure: Files and folders in a sample package of ROS

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

ROS packages

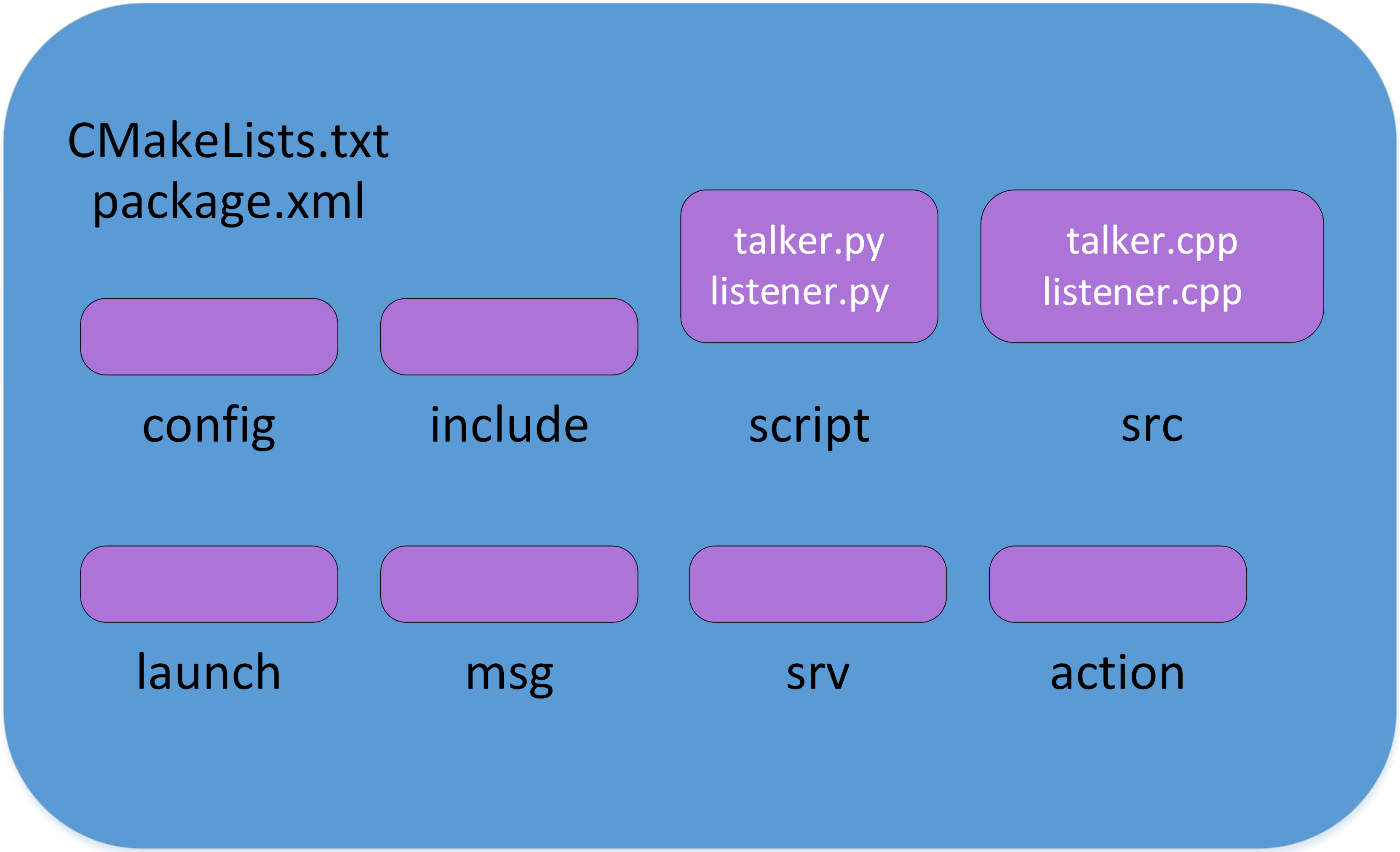


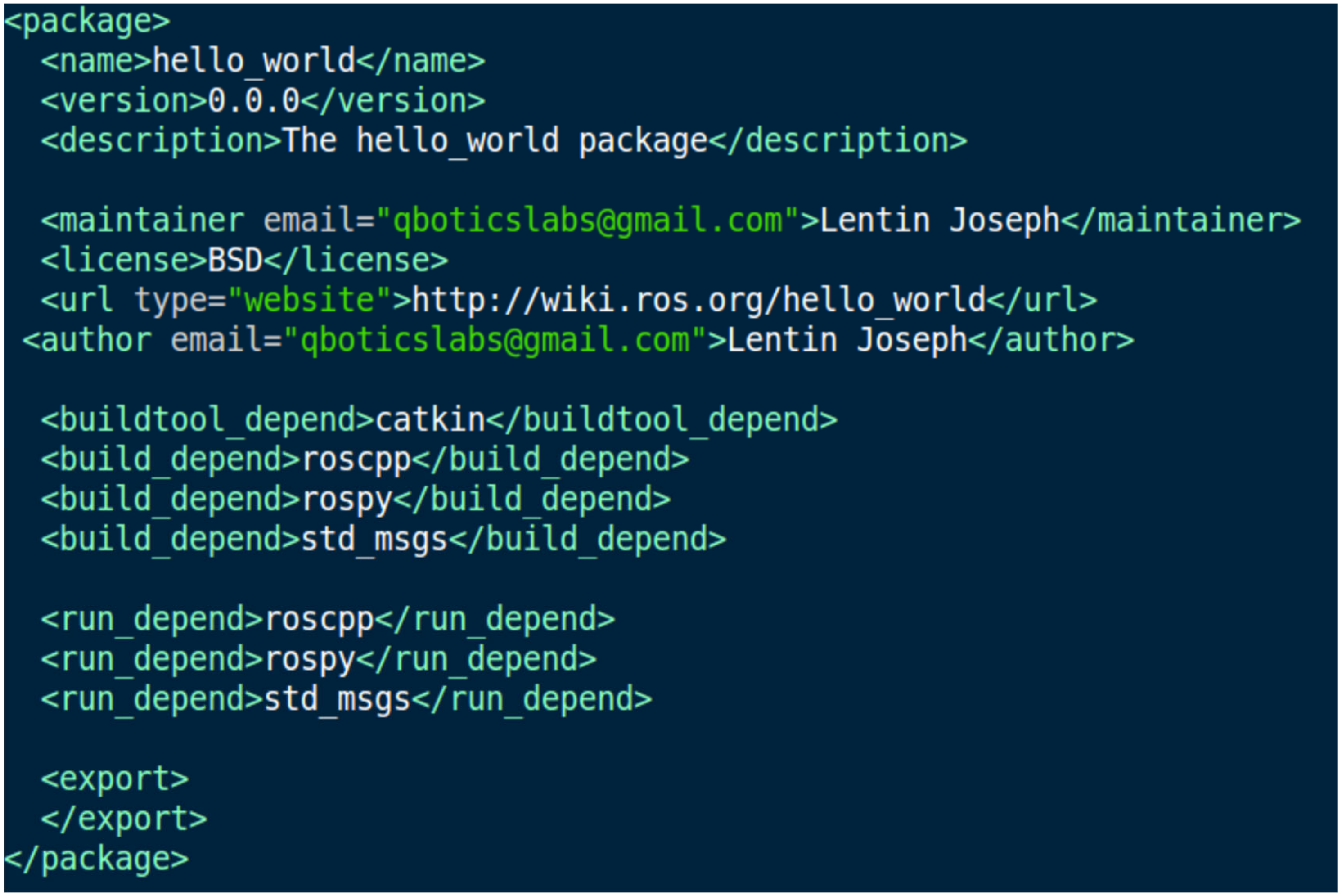
Figure: Structure of a typical ROS package

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

ROSbash

- Some of the rosbash commands are:
  - roscd
  - roscp
  - rosed
  - rosrun

package.xml



```
<package>
  <name>hello_world</name>
  <version>0.0.0</version>
  <description>The hello_world package</description>

  <maintainer email="qboticslabs@gmail.com">Lentin Joseph</maintainer>
  <license>BSD</license>
  <url type="website">http://wiki.ros.org/hello_world</url>
 <author email="qboticslabs@gmail.com">Lentin Joseph</author>

  <buildtool_depend>catkin</buildtool_depend>
  <build_depend>roscpp</build_depend>
  <build_depend>rospy</build_depend>
  <build_depend>std_msgs</build_depend>

  <run_depend>roscpp</run_depend>
  <run_depend>rospy</run_depend>
  <run_depend>std_msgs</run_depend>

  <export>
  </export>
</package>
```

Figure: Structure of a typical ROS package

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

ROS messages

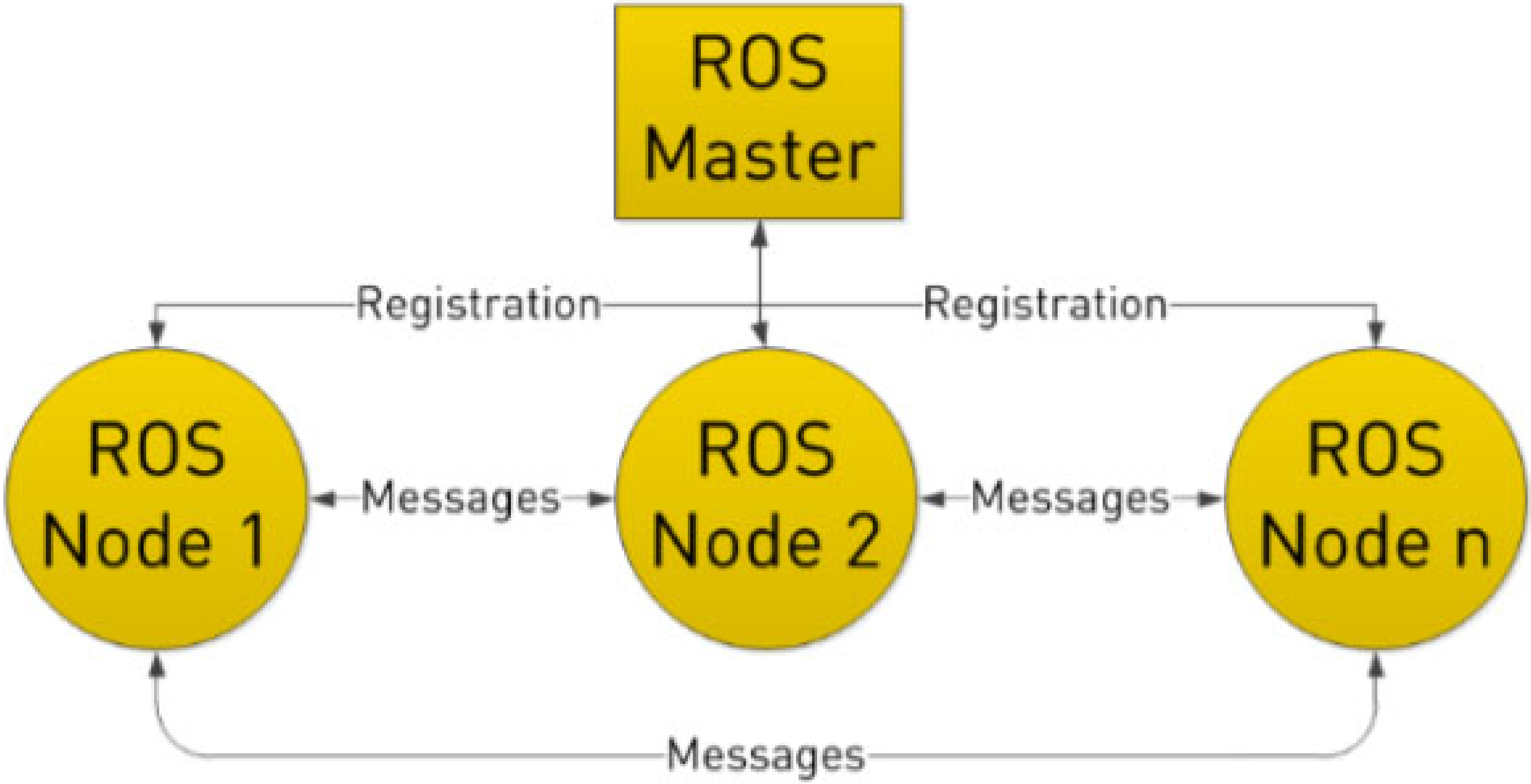


Figure: Ros messages

Source: http://www2.ece.ohiostate.edu/~zhang/RoboticsClass/docs/ECE5463_ROSTutorialLecture1.pdf

ROS services

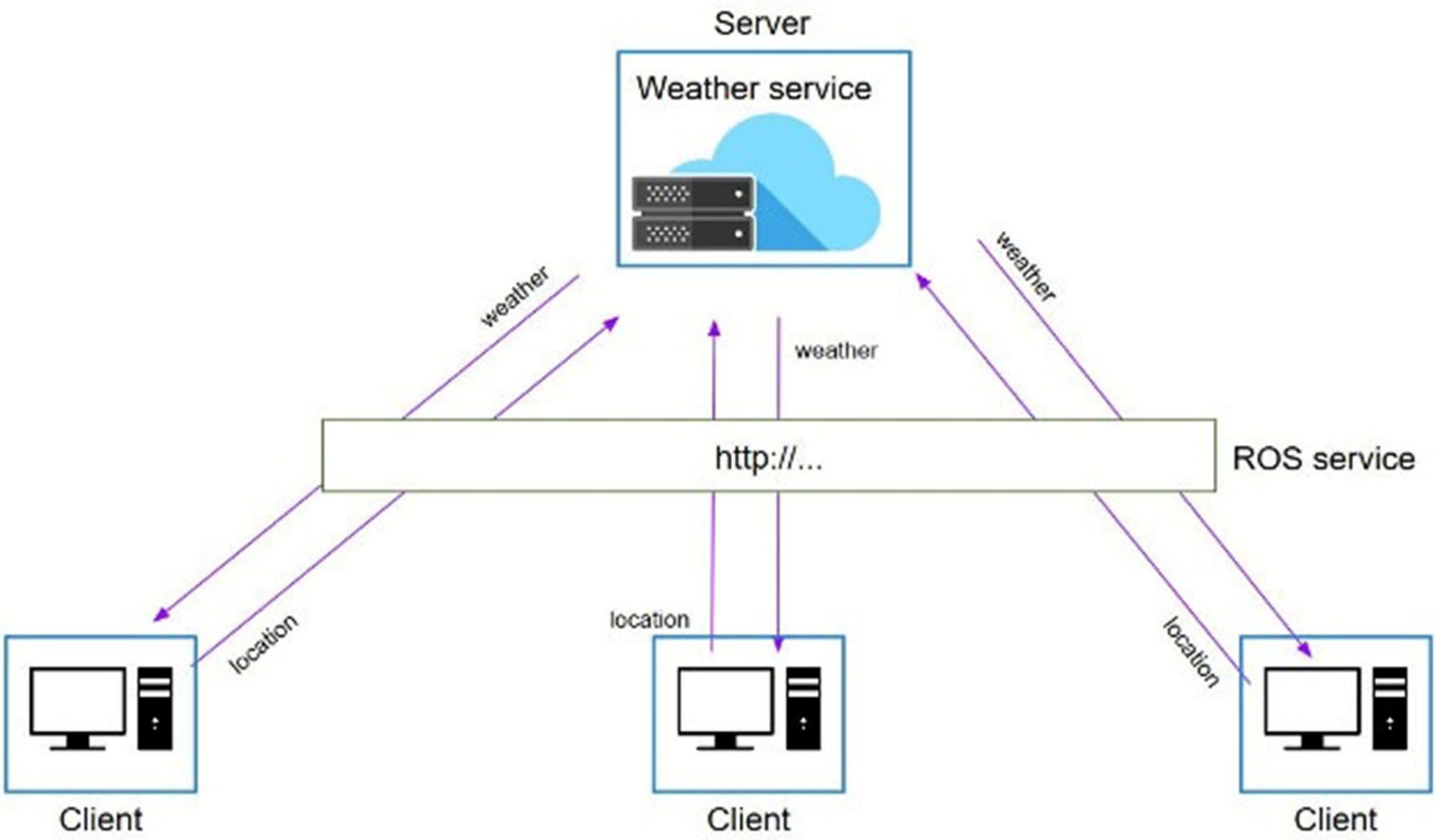


Figure: Ros Services
Source: https://roboticsbackend.com/what-is-a-ros-service/

The computational graph level (1 of 2)

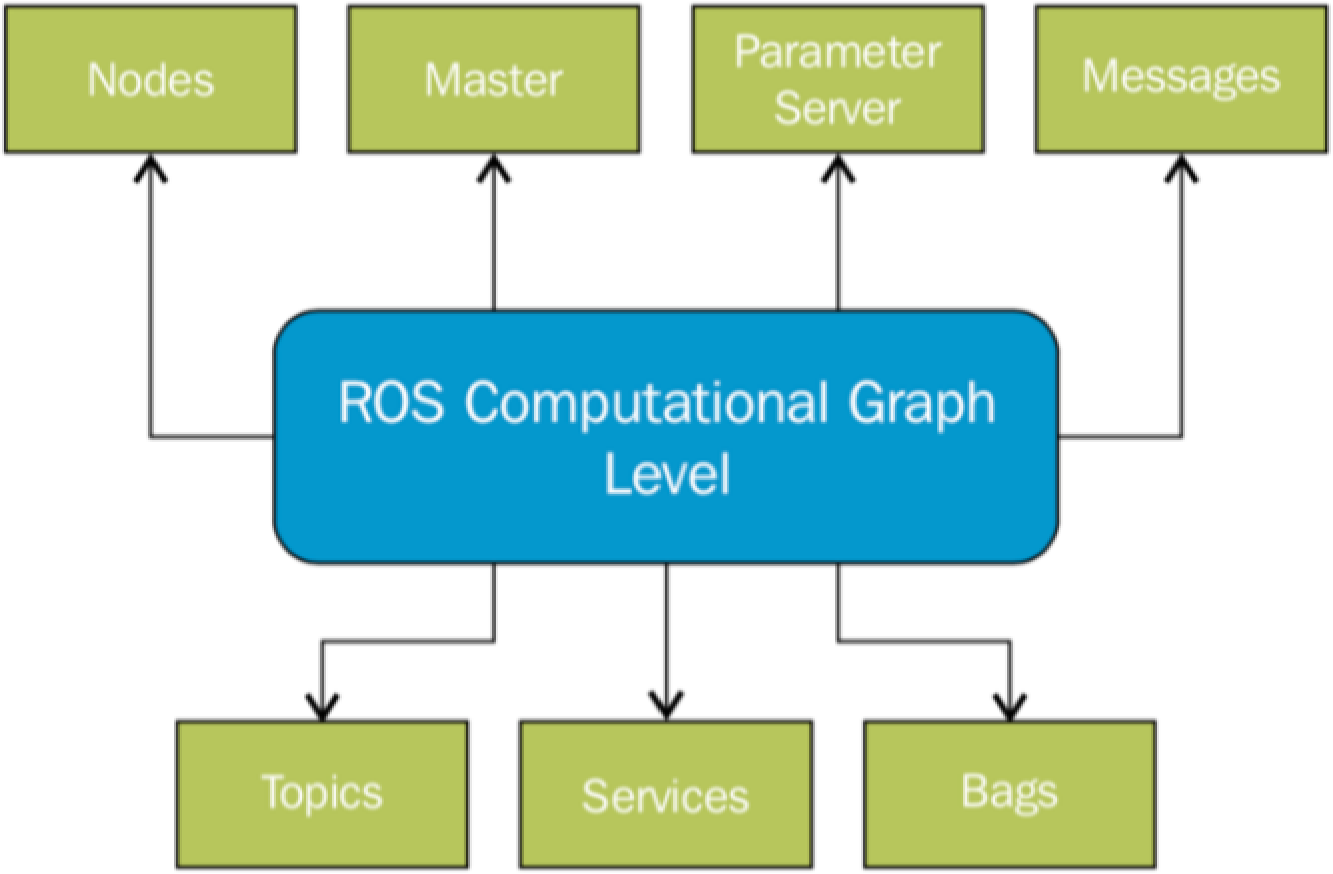


Figure: ROS Computational graph level

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

The computational graph level (2 of 2)

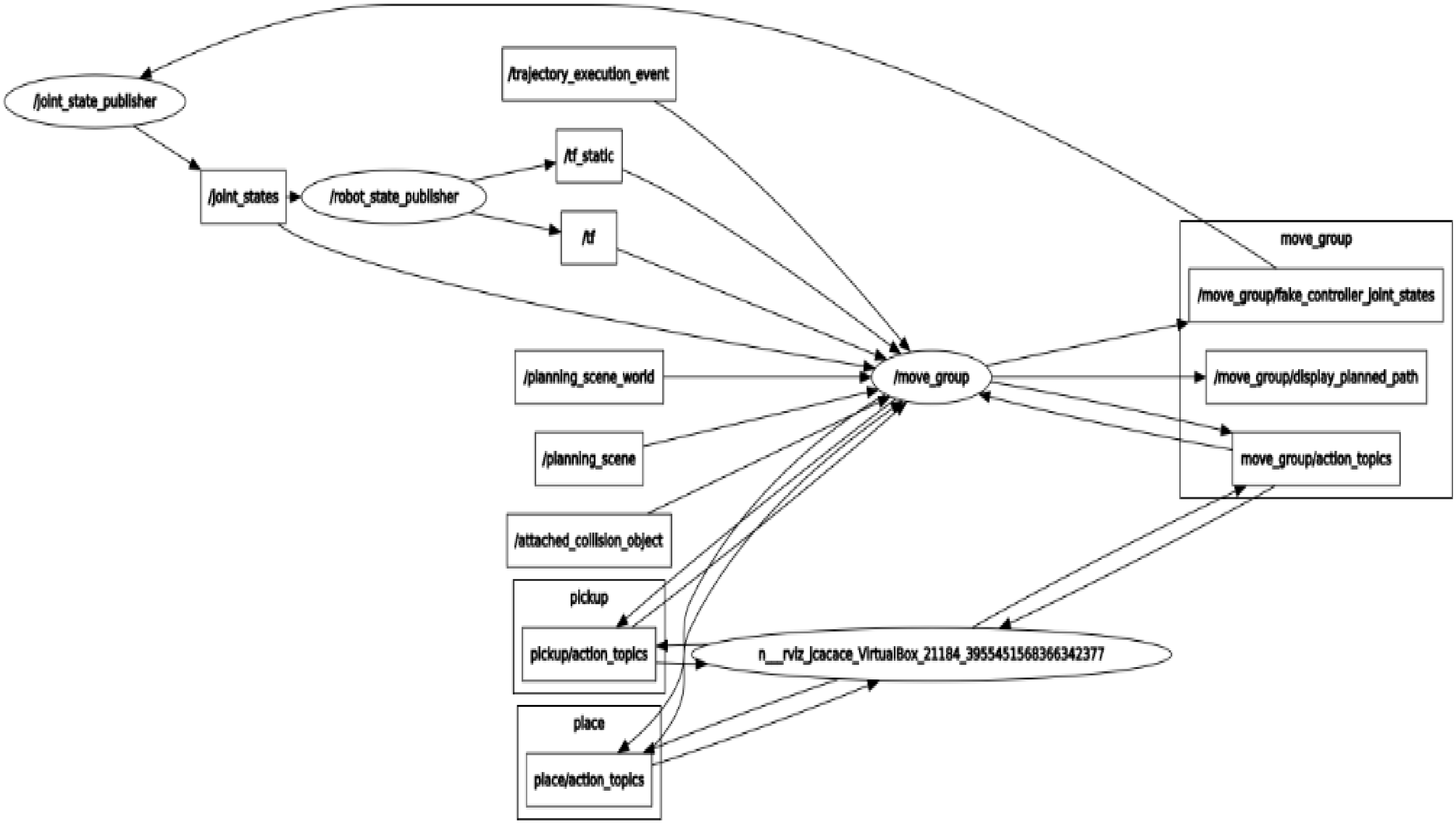


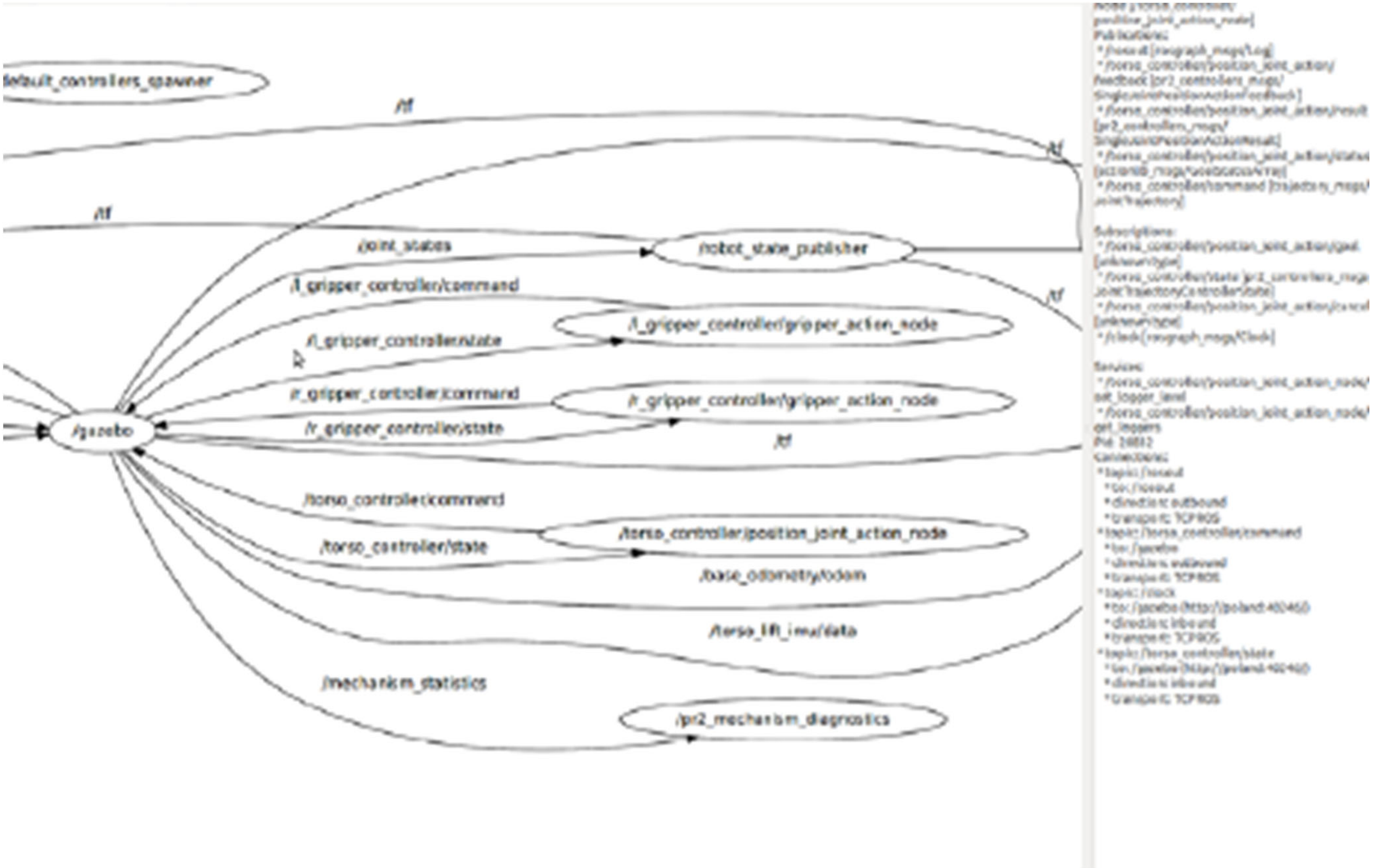
Figure: Graph of Communication using Topics

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

The community level

- Various resources in these communities are as follows:
  - Distributions
  - Repositories
  - ROS Wiki
  - Bug ticket system
  - Mailing lists
  - Blog

Debugging and visualization (1 of 4)


default_controllers_spawner
/tf
/tf
/tf
/joint_states
/robot_state_publisher
/l_gripper_controller/command
/tf
/l_gripper_controller/gripper_action_node
/l_gripper_controller/state
/r_gripper_controller/command
/r_gripper_controller/gripper_action_node
/gazebo
/r_gripper_controller/state
/tf
/torso_controller/command
/torso_controller/position_joint_action_node
/torso_controller/state
/base_odometry/odom
/torso_lift_imu/data
/mechanism_statistics
/pr2_mechanism_diagnostics

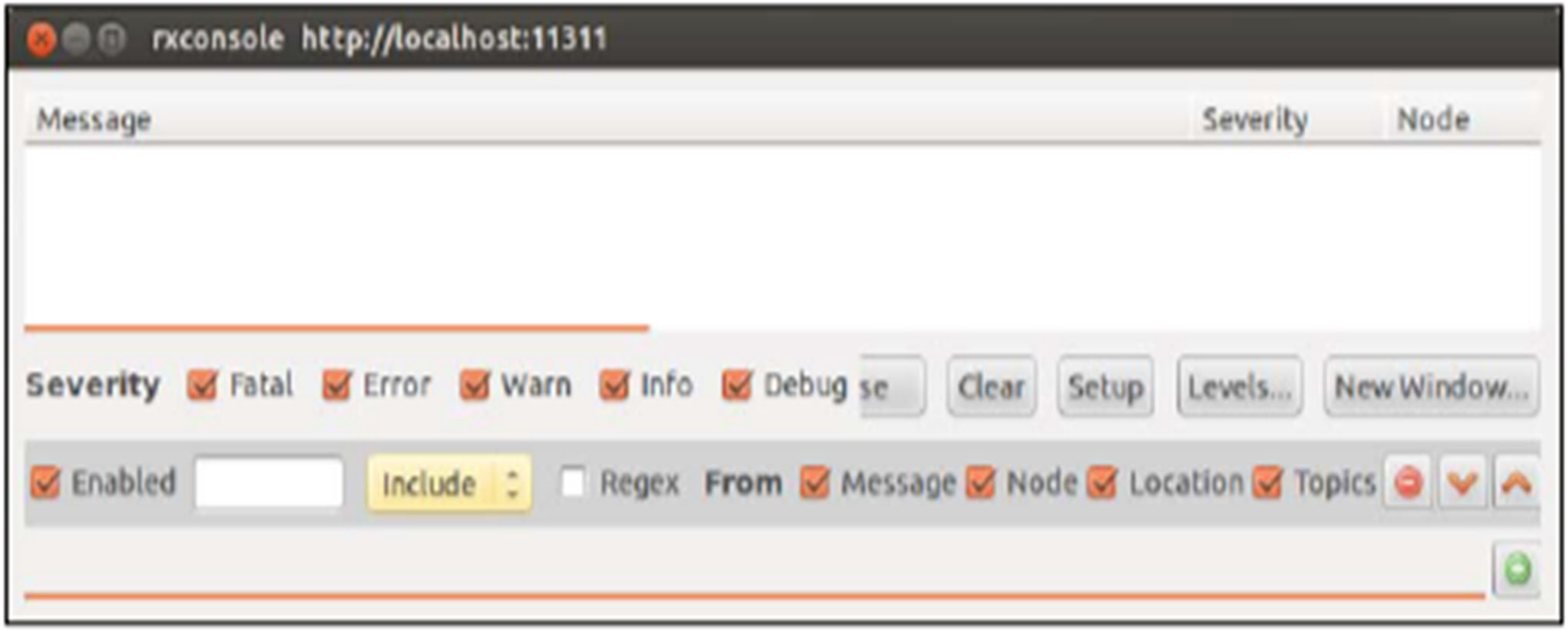
```
Node [/torso_controller/
position_joint_action_node]
Publications:
 * /rosout [rosgraph_msgs/Log]
 * /torso_controller/position_joint_action/
feedback [pr2_controllers_msgs/
SingleJointPositionActionFeedback]
 * /torso_controller/position_joint_action/result
[pr2_controllers_msgs/
SingleJointPositionActionResult]
 * /torso_controller/position_joint_action/status
[actionlib_msgs/GoalStatusArray]
 * /torso_controller/command [trajectory_msgs/
JointTrajectory]

Subscriptions:
 * /torso_controller/position_joint_action/goal
[unknown type]
 * /torso_controller/state [pr2_controllers_msgs/
JointTrajectoryControllerState]
 * /torso_controller/position_joint_action/cancel
[unknown type]
 * /clock [rosgraph_msgs/Clock]

Services:
 * /torso_controller/position_joint_action_node/
set_logger_level
 * /torso_controller/position_joint_action_node/
get_loggers
Pid: 16812
Connections:
 * topic: /rosout
    * to: /rosout
    * direction: outbound
    * transport: TCPROS
 * topic: /torso_controller/command
    * to: /gazebo
    * direction: outbound
    * transport: TCPROS
 * topic: /clock
    * to: /gazebo (http://poland:40246/)
    * direction: inbound
    * transport: TCPROS
 * topic: /torso_controller/state
    * to: /gazebo (http://poland:40246/)
    * direction: inbound
    * transport: TCPROS
```

Figure: Debugging and visualization graph

Source: joseph, L. 2015. Mastering ROS for robotics programming. Birmingham: packt publishing ltd.

Debugging and visualization (2 of 4)

- Plotting Tools
  - Time series plots
  - Image visualization tools
  - 3D visualization tools



Severity ☑ Fatal ☑ Error ☑ Warn ☑ Info ☑ Debug

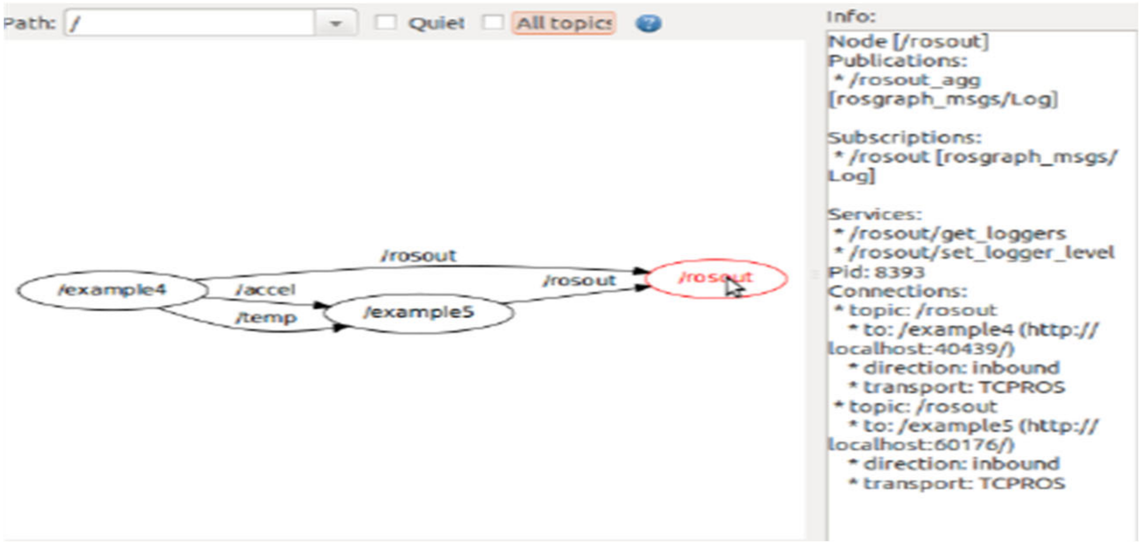
☑ Enabled □ Regex From ☑ Message ☑ Node ☑ Location ☑ Topics

Figure: screenshot of rxconsole

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

Debugging and visualization (3 of 4)

IBM ICE (Innovation Centre for Education)



```
Info:
Node [/rosout]
Publications:
 * /rosout_agg
[rosgraph_msgs/Log]

Subscriptions:
 * /rosout [rosgraph_msgs/
Log]

Services:
 * /rosout/get_loggers
 * /rosout/set_logger_level
Pid: 8393
Connections:
 * topic: /rosout
    * to: /example4 (http://
localhost:40439/)
    * direction: inbound
    * transport: TCPROS
 * topic: /rosout
    * to: /example5 (http://
localhost:60176/)
    * direction: inbound
    * transport: TCPROS
```

Figure: Node’s graph online – rxgraph

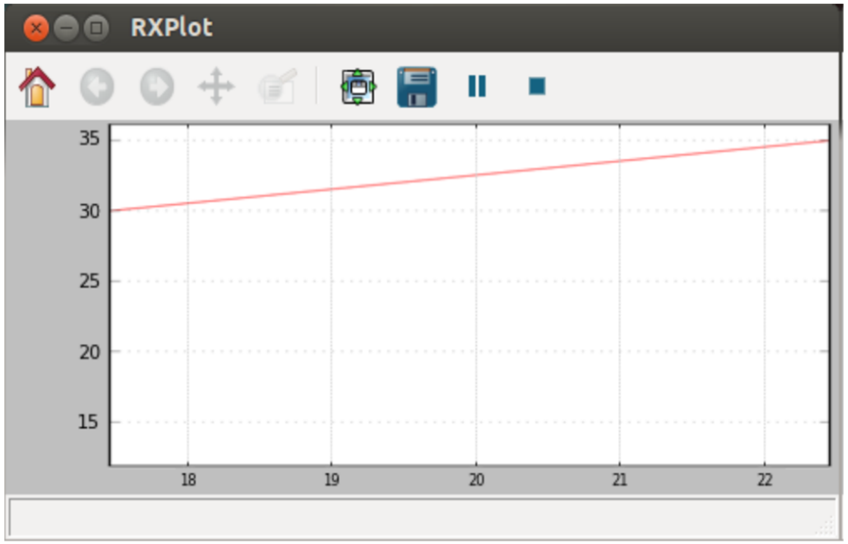
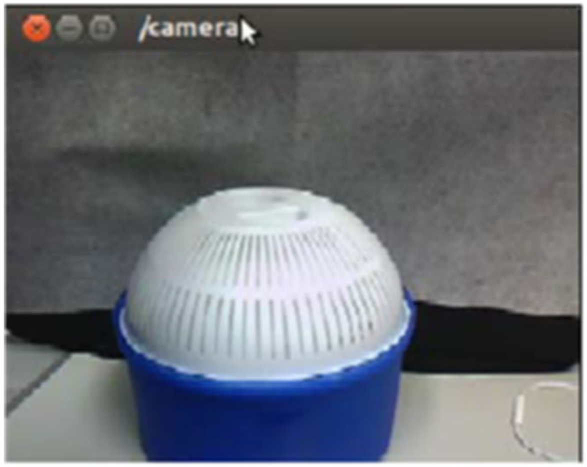


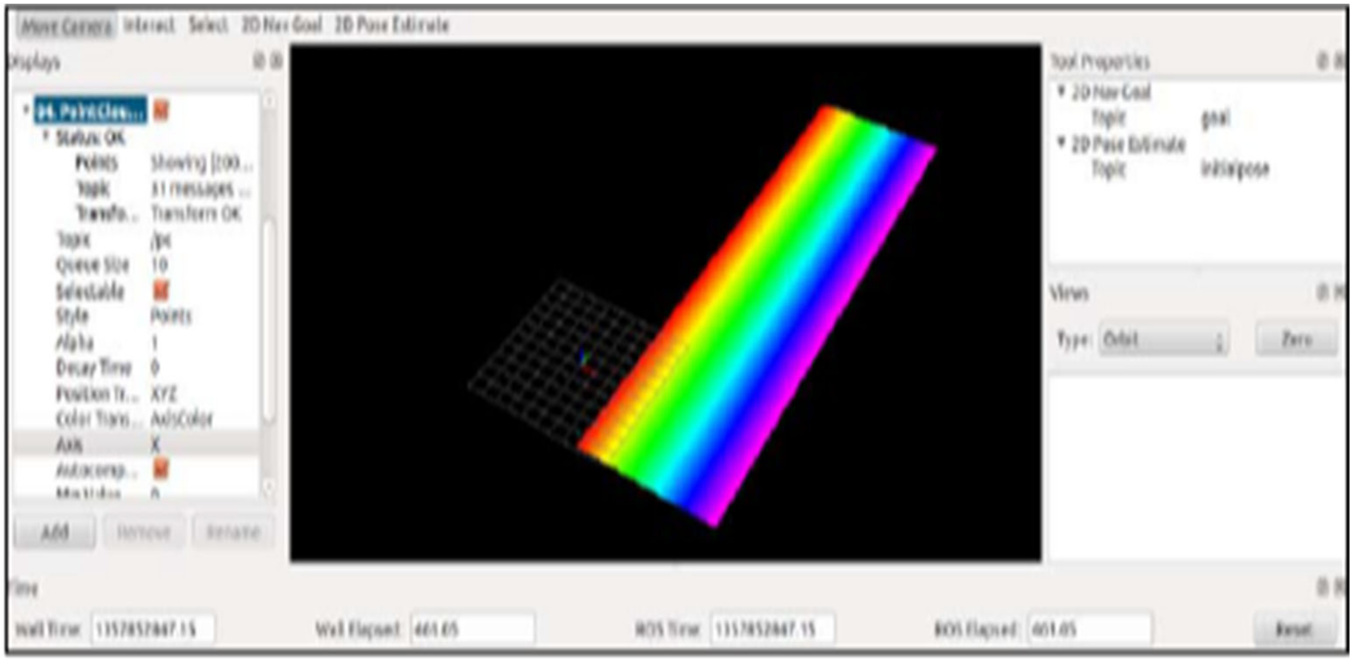
Figure: screenshot of rxplot

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

Debugging and visualization (4 of 4)

- Visualization of images





PointCloud... ☑
Status: OK
Points Showing [200...
Topic 31 messages ...
Transfo... Transform OK
Topic /pc
Queue Size 10
Selectable ☑
Style Points
Alpha 1
Decay Time 0
Position Tr... XYZ
Color Trans... AxisColor
Axis X
Autocomp... ☑

2D Nav Goal
Topic goal
2D Pose Estimate
Topic initialpose

Type: Orbit

Wall Time: 1357852847.15 Wall Elapsed: 461.05 ROS Time: 1357852847.15 ROS Elapsed: 461.05

screenshot of output of a camera as a camera topic

screenshot of output of rviz

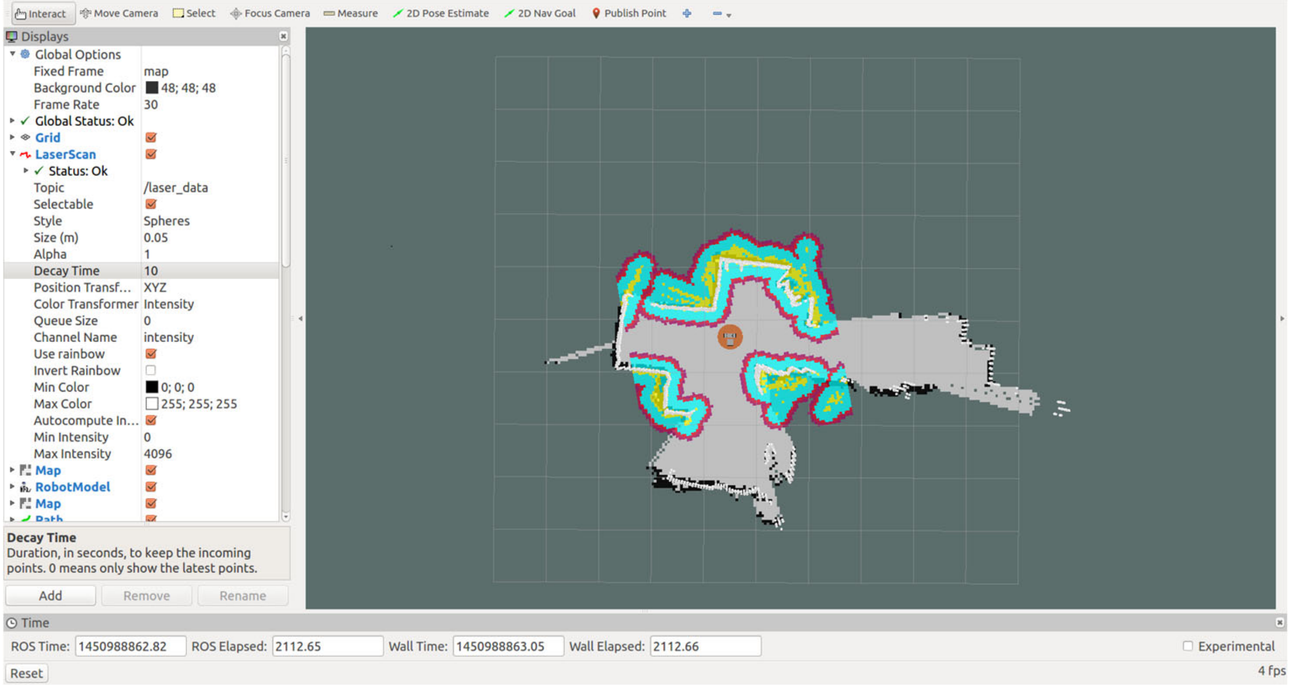
Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

Using sensors and actuators (1 of 3)

- Robots must sense the environment around them in order to react to variations in tasks. The sensors can range from very simple minimal setup designed for quick installation to highly complex and expensive sensor setups. For example :
  - Visual cameras
  - Depth cameras

Using sensors and actuators (2 of 3)

Laser scanners



Global Options	
Fixed Frame	map
Background Color	■ 48; 48; 48
Frame Rate	30
✓ Global Status: Ok	
Grid	☑
LaserScan	☑
✓ Status: Ok	
Topic	/laser_data
Selectable	☑
Style	Spheres
Size (m)	0.05
Alpha	1
Decay Time	10
Position Transf...	XYZ
Color Transformer	Intensity
Queue Size	0
Channel Name	intensity
Use rainbow	☑
Invert Rainbow	□
Min Color	■ 0; 0; 0
Max Color	□ 255; 255; 255
Autocompute In...	☑
Min Intensity	0
Max Intensity	4096
Map	☑
RobotModel	☑
Map	☑
Path	☑

Decay Time
Duration, in seconds, to keep the incoming points. 0 means only show the latest points.

ROS Time: 1450988862.82 ROS Elapsed: 2112.65 Wall Time: 1450988863.05 Wall Elapsed: 2112.66 □ Experimental

Figure: Screenshot of Laser Scanner in the Simulator

Source: https://answers.ros.org/question/231560/smearing-ghosting-of-laser-scan-in-move_base-costmap/

Using sensors and actuators (3 of 3)

Shaft Encoders

Figure: Rotary Encoder

Source: http://www.robo-dyne.com/shop/rotary-encoder-illuminated-redgreen/?lang=en

3D modeling and simulation (1 of 2)

Stage

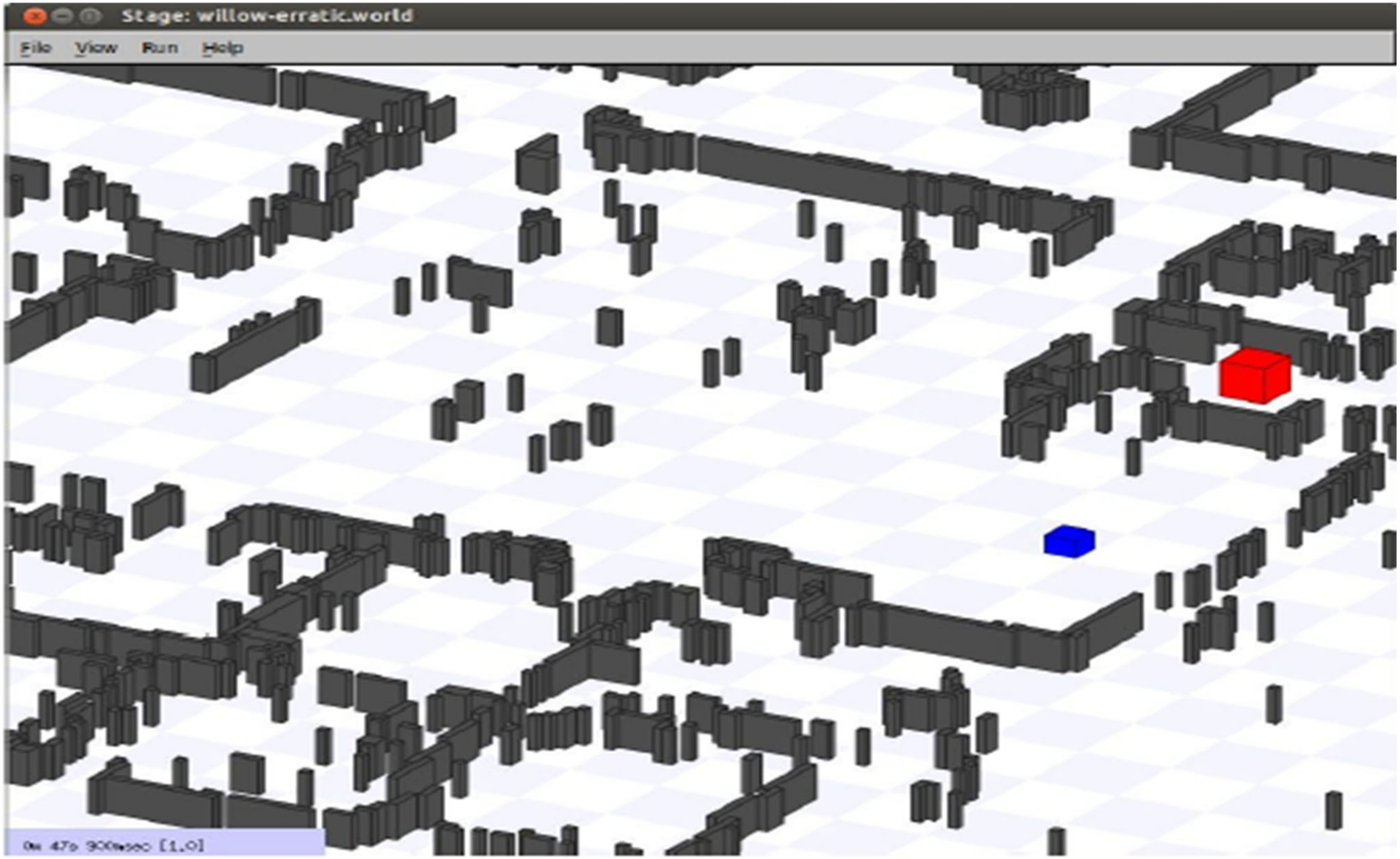


Figure: Screenshot of stage simulator

Source: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

3D modeling and simulation (2 of 2)

Gazebo

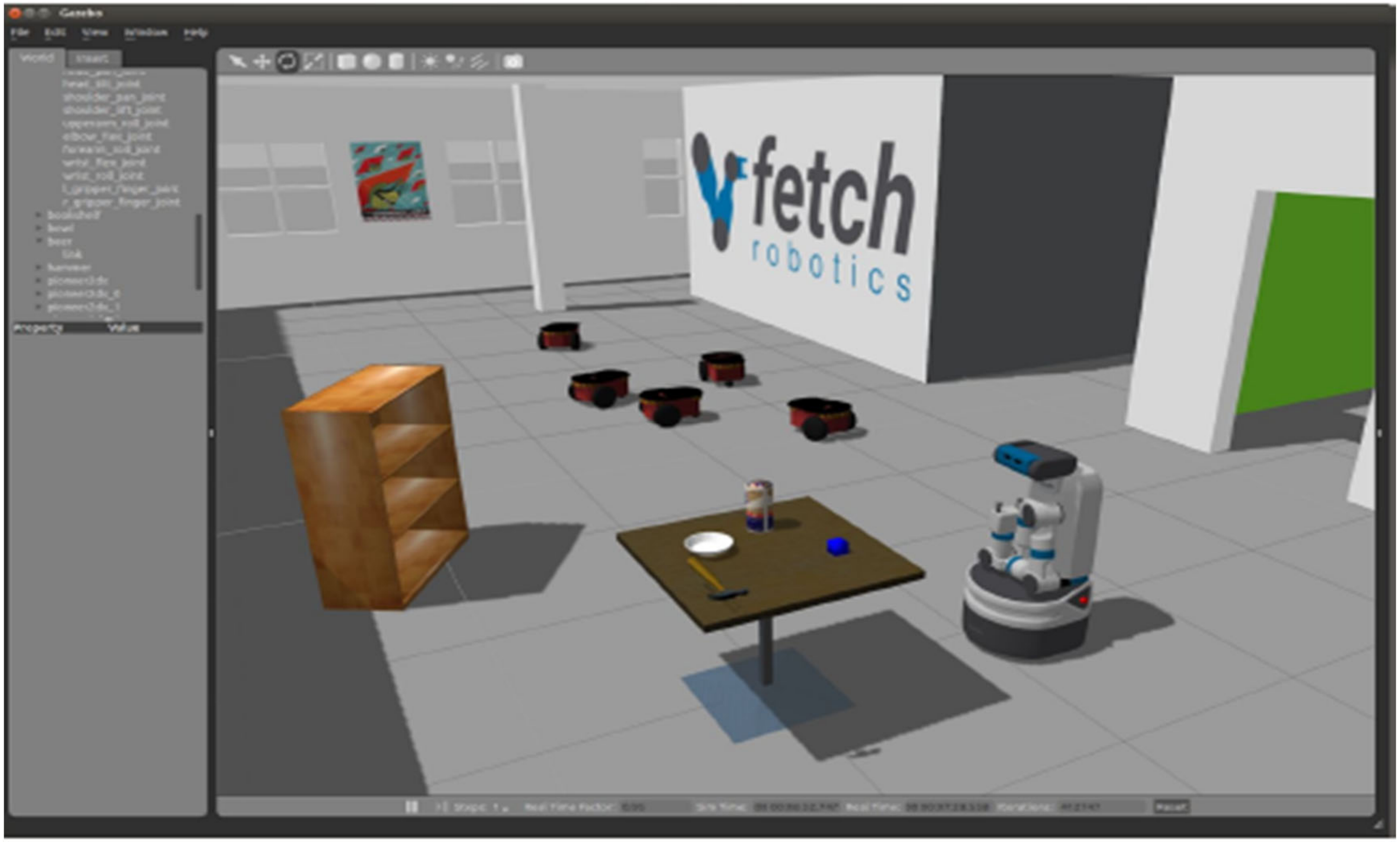


Figure: Screenshot of Gazebo simulator

Figure: Joseph, L. 2015. Mastering ROS for Robotics Programming. Birmingham: Packt Publishing Ltd.

Computer vision

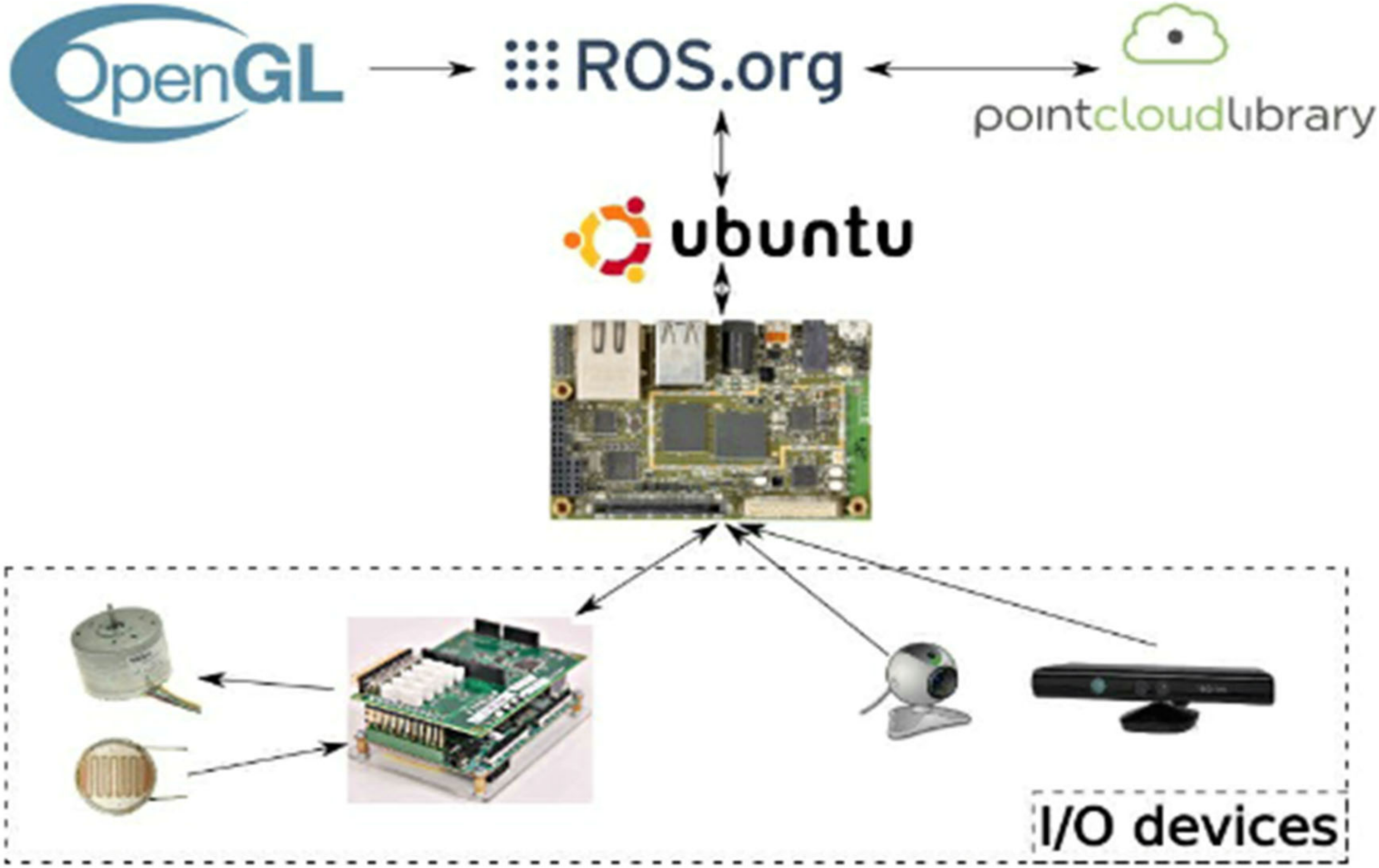


Figure: Computer Vision and ROS

Source: https://www.inforcecomputing.com/robots_hearts_are_beating_with_inforce_platforms.)

Checkpoint (1 of 2)

Multiple choice questions:

1. Which of these is not a widely used simulator?
   1. a) Arbotix
   2. b) Stage
   3. c) Maxwell
   4. d) Gazebo

2. Robot operating system (ROS) is an _____________ platform.
   1. a) Framework Development
   2. b) Application Development
   3. c) Interaction Design (ID)
   4. d) None of the above

3. ROS has got tools for
   1. a) Simulation
   2. b) Visualization
   3. c) Debugging
   4. d) All of the above

Checkpoint solutions(1 of 2)

Multiple choice questions:

1. Which of these is not a widely used simulator?
   a) Arbotix
   b) Stage
   c) Maxwell
   d) Gazebo

2. Robot operating system (ROS) is an ____________ platform.
   a) Framework Development
   b) Application Development
   c) Interaction Design (ID)
   d) None of the above

3. ROS has got tools for
   a) Simulation
   b) Visualization
   c) Debugging
   d) All of the above

Checkpoint (2 of 2)

Fill in the blanks:

1. ROS packages are maintained using ____________.
2. __________are the process that perform computation.
3. Scalar Values can be plotted using _________.
4. ________ is one of the 3D visualization tool.

True or False:

1. Actuators like Dynamixel servos are also supported in ROS . True/False
2. There is a package named MoveItall for robot motion planning. True/False
3. Modeling in ROS is performed using URDF. True/False

Checkpoint solutions (2 of 2)

Fill in the blanks:

1. ROS packages are maintained using ______**VCS**______
2. ______**Nodes**______ are the process that perform computation
3. Scalar Values can be plotted using ____**Time series plot**______
4. ____**rviz**_____ is one of the 3D visualization tool.

True or False:

1. Actuators like Dynamixel servos are also supported in ROS . **True**
2. There is a package named MoveItall for robot motion planning. **False**
3. Modeling in ROS is performed using URDF. **True**

Question bank

Two mark questions:

1. What is ROS?
2. Which are the various robotic platforms?
3. What is visualization graph?
4. Name the components of graph layer?

Four mark questions:

1. Explain the file system level.
2. Describe the support for OpenCV in ROS .
3. Explain the concept of camera resolutions.
4. Explain gazebo simulator.

Eight mark questions:

1. Explain the various debugging and visualization options in ROS.
2. Explain the different types sensors and actuators.

Unit summary

Having completed this unit, you should be able to:

- Understand the concept of Robot Operating System
- Gain an insight into various levels of ROS package
- Gain knowledge on debugging and Visualization in ROS
- Understand the interaction of computer vision and ROS