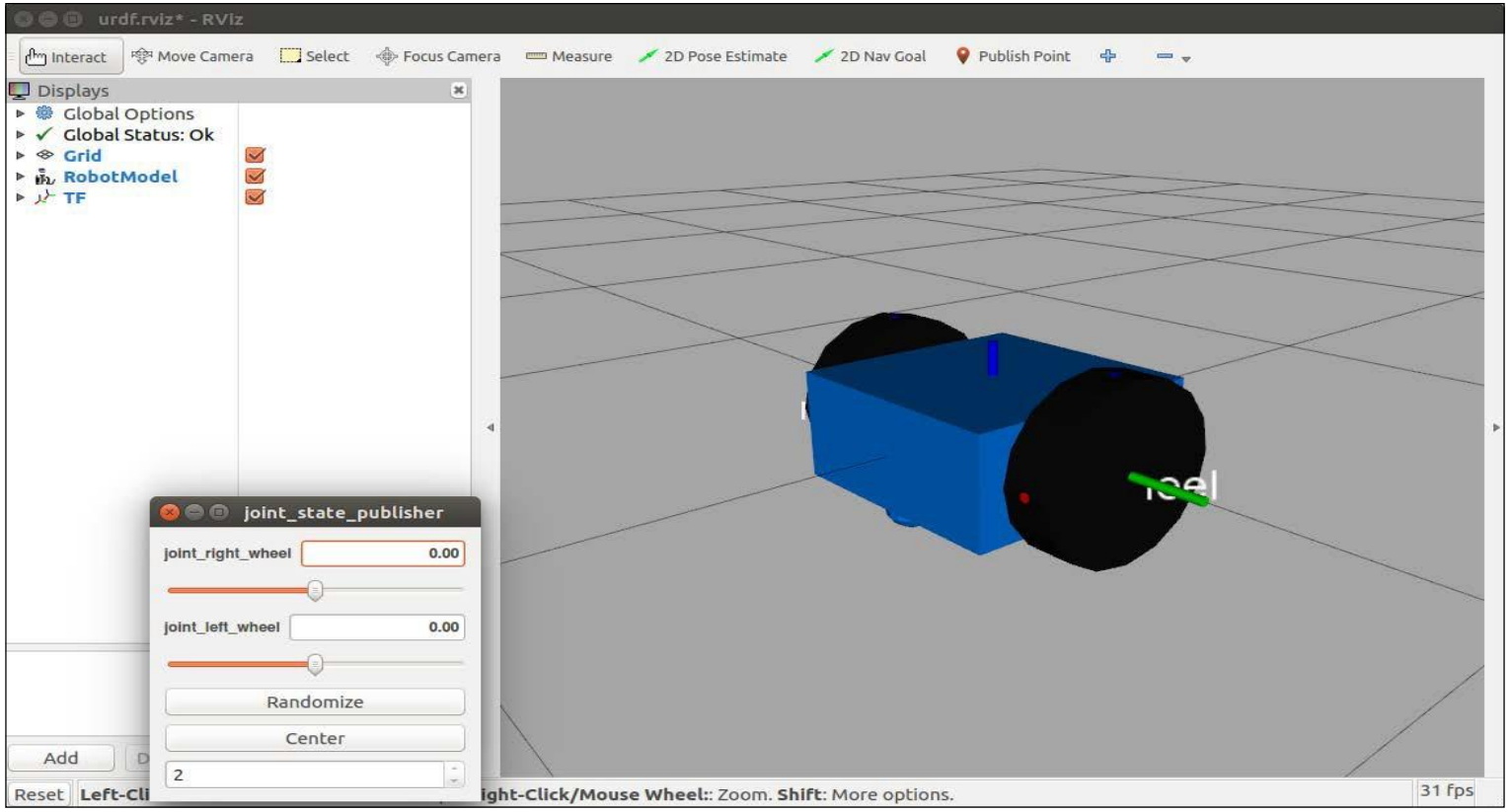
CENG 5434 ROS PACKAGE,
ROS WORKSPACE,
CHAPTER 2- RVIZ, URDF, GAZEBO



Tutorial Outline

- **Installing ROS Packages**
- **Creating Your Workspace**
- **CHAPTER 2 IN TEXTBOOK**

Installing and launching ROS

CHAPTER 1 PG 5-9

ROS Robotics By Example
Second Edition

Recall: What is a package?

- All the files that a specific ROS program contains; all its cpp files, python files, configuration files, compilation files, launch files, and parameters files.

- Generally all those files in the package are organized with the following structure:
  - **launch** folder: Contains launch files
  - **src** folder: Source files (cpp, python)

  } Not always

  - **CMakeLists.txt**: List of cmake rules for compilation
  - **package.xml**: Package information and dependencies

Installing ROS package

Using Ubuntu´s package manager

•The **apt-get** command is a powerful command-line tool, which works with Ubuntu's Advanced Packaging Tool (APT) performing such functions as downloading and installation of new software packages, updating, etc.

•This method is used for installing **released ROS packages** (packages verified by ROS developers) and install the package's necessary dependencies.

• Command:

```
sudo apt-get install ros-<ros_distro>-<package-name>
```

•Example

```
sudo apt-get install ros-kinetic-urdf
```

If you wish to install the ROS Kinetic source code and build the software, refer to the instructions at http://wiki.ros.org/kinetic/Installation/Source. The instructions presented here to install ROS Kinetic with **Debian** packages can also be found at http://wiki.ros.org/kinetic/Installation/Ubuntu.

Page 5

http://wiki.ros.org/kinetic/Installation/Ubuntu

SOURCES.LIST, KEYS, KINETIC-DESKTOP- FULL (PAGE 6)

```
$ sudo sh -c 'echo "deb
http://packages.ros.org/ros/ubuntu
$(lsb_release -sc) main" > /etc/apt/sources.list.d/ros-
latest.list'

$ sudo apt-key adv --keyserver hkp://ha.pool.sks-
keyservers.net:80
--recv-key 421C365BD9FF1F717815A3895523BAEEB01FA116

$ sudo apt-get update
```

$ sudo apt-get install ros-kinetic-desktop-full

Initialize rosdep

The ROS system may depend on software packages that are not loaded initially. These software packages external to ROS are provided by the operating system. The ROS environment command `rosdep` is used to download and install these external packages. Type the following commands:

```
$ sudo rosdep init
$ rosdep update
```

Environment setup

Your terminal session must now be made aware of these ROS files so that it knows what to do when you attempt to execute ROS command-line commands. Running this script will set up the ROS environment variables:

```
$ source /opt/ros/kinetic/setup.bash
```

Alternatively, it is convenient if the ROS environment variables are automatically added to your terminal session every time a new shell is launched. If you are using bash for your terminal shell, do this by typing the following commands:

```
$ echo "source /opt/ros/kinetic/setup.bash" >> ~/.bashrc
$ source ~/.bashrc
```

Now when a new terminal session is launched, the bash shell is automatically aware of the ROS environment variables.

$ sudo apt-get install python-rosinstall

Installing ROS package – FROM SOURCE

Installing from source code

•If the package is not released, you will need to install it from source code. Find out where the code is hosted (mostly github), **and install the package in the src folder from your workspace directory** `(~/catkin_ws/src)`

https://github.com/ros/urdf, https://github.com/ROBOTIS-GIT/turtlebot3

- Command:

```
git clone <address> / git clone -b <branch> <address>
```

- Example:

```
$ cd ~/catkin_ws/src
$ git clone -b kinetic-devel https://github.com/ros/urdf.git
$ git clone https://github.com/ROBOTIS-GIT/turtlebot3
$ cd ~/catkin_ws
$ catkin_make
```

Installing ROS package

Installing from source code

• Problem: You have to take care of the dependencies by yourself.

• Whenever you see "Could not find a configuration file for package <package_name>" it means this package is missing and is needed for compiling your code.

•Possible solution: Use **rosdep** command, which will automatically try to find all the dependencies of your package and install them.

IMPORTANT NOTE

•Sometimes the `<package-name> or <address>` arguments correspond to a directory which contains more than one package.

YOUR WORKSPACE

Creating a catkin workspace

The next step is to create a catkin workspace. A catkin workspace is a directory (folder) in which you can create or modify existing catkin packages. The catkin structure simplifies the build and installation process for your ROS packages. The ROS wiki website is http://wiki.ros.org/catkin/Tutorials/create_a_workspace.

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace

$ cd ~/catkin_ws/
$ catkin_make
```

Creating a catkin workspace REVIEW

(Chapter 1, page 9)

```
$ mkdir -p ~/catkin_ws/src
$ cd ~/catkin_ws/src
$ catkin_init_workspace
$ cd ~/catkin_ws/
$ catkin_make

$ echo "source ~/catkin_ws/devel/setup.bash"
>> ~/.bashrc
$ source ~/.bashrc
$ echo $ROS_PACKAGE_PATH    (CHECK IT)
```

Creating a ROS package

(Chapter 2, page 41)

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg ros_robotics
$ cd ~/catkin_ws
$ catkin_make
```

MAKE SURE catkin_ws is sourced for each terminal or in .bashrc

Source the ~/catkin_ws in shell and .bashrc

harman@harman-VirtualBox:~/catkin_ws$ echo "source ~/catkin_ws/devel/setup.bash" >> ~/.bashrc

harman@harman-VirtualBox:~/catkin_ws$ source ~/.bashrc

Echo the ROS Path for ROS and ~/catkin_ws

harman@harman-VirtualBox:~/catkin_ws$ echo $ROS_PACKAGE_PATH

/home/harman/catkin_ws/src:/opt/ros/kinetic/share (ROS was previously sourced)

Initial Files After catkin_make

harman@harman-VirtualBox:~/catkin_ws$ ls

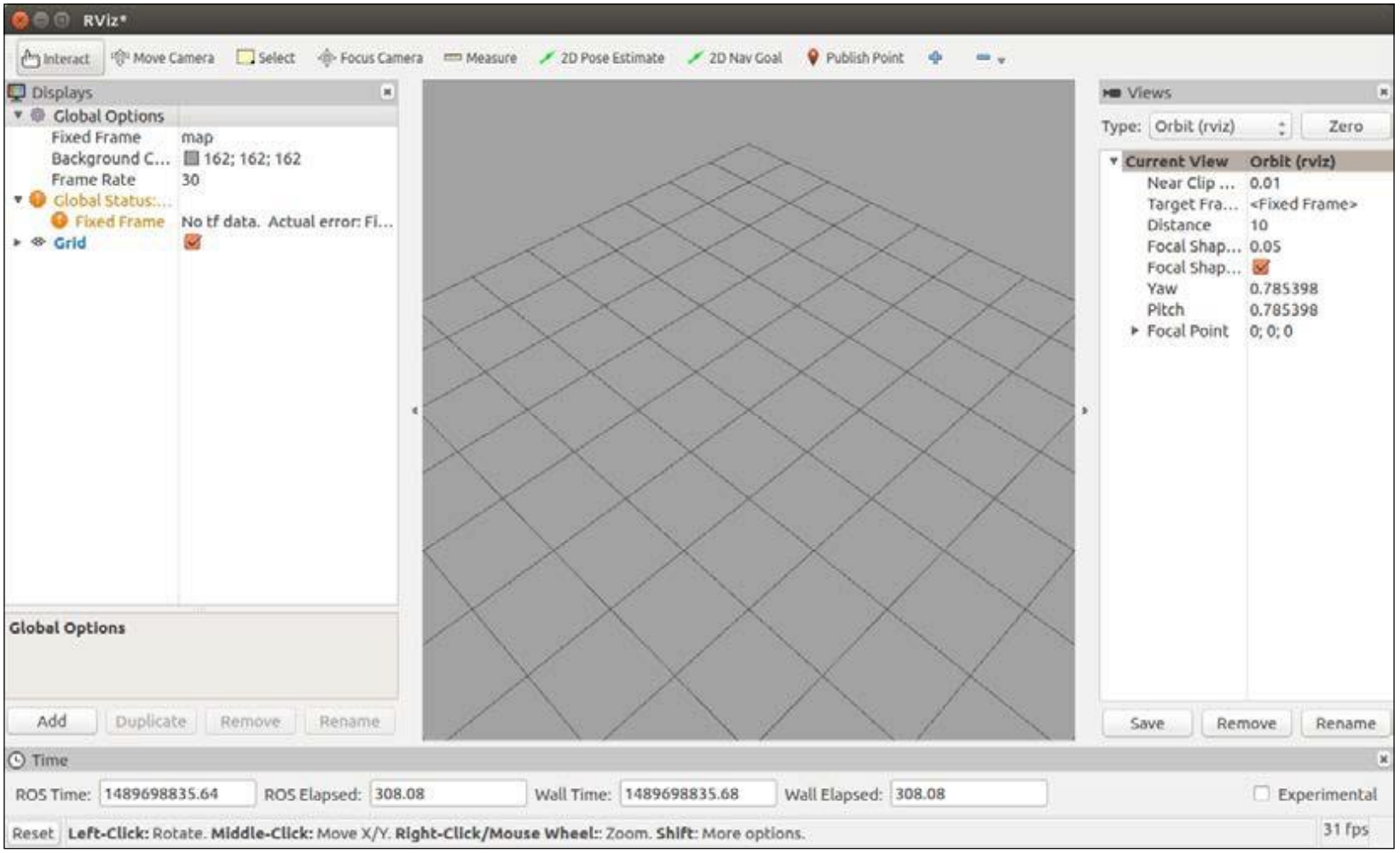
build devel src

harman@harman-VirtualBox:~/catkin_ws$ cd src

harman@harman-VirtualBox:~/catkin_ws/src$ ls

CmakeLists.txt

RVIZ for Visualization



URDF - XACRO files (XML)

The Universal Robotic Description Format (URDF) is an XML file format used in ROS to describe all elements of a robot.

To use a URDF file in Gazebo, some additional simulation-specific tags must be added to work properly with Gazebo.

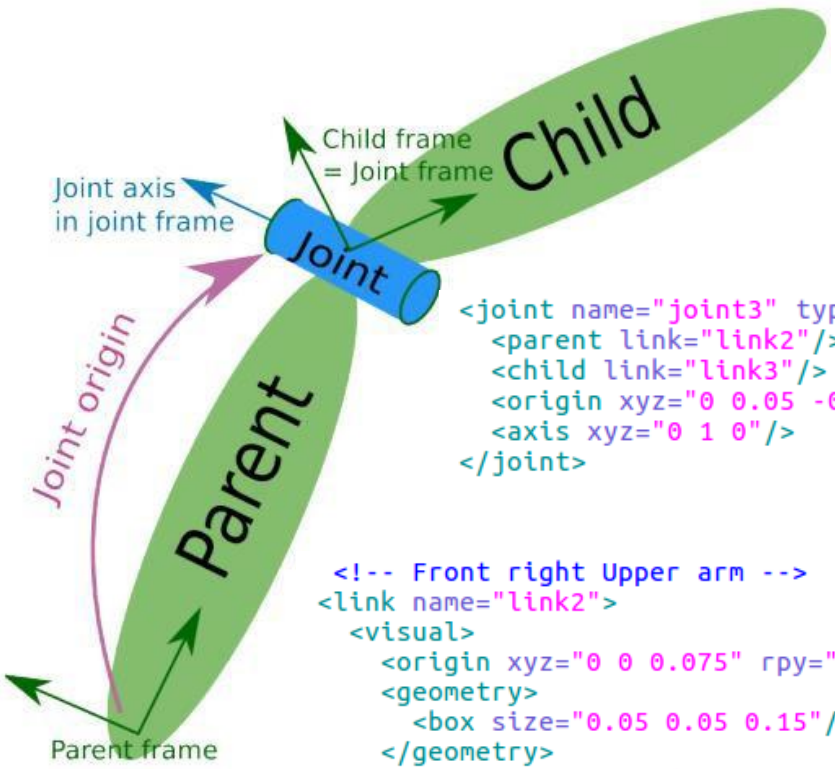
URDF

- Specify the kinematic and dynamic properties
- Tags: link, joint, transmission
- Order in the file does not matter

XACRO

- XML Macro Language used for URDF simplification
- Reduce redundancy and increase modularity
- Use parametrization (Use parameters for lengths and links and math for origin and inertia calculation)

Link and joint representation




```
 <!-- Front right End arm-->
<link name="link3">
  <visual>
    <origin xyz="0 0 -0.1" rpy="0 0 0"/>
    <geometry>
      <box size="0.025 0.025 0.025"/>
    </geometry>
    <material name="orange"/>
  </visual>
</link>
```

```
<joint name="joint3" type="continuous">
  <parent link="link2"/>
  <child link="link3"/>
  <origin xyz="0 0.05 -0.07" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
</joint>
```

```
 <!-- Front right Upper arm -->
<link name="link2">
  <visual>
    <origin xyz="0 0 0.075" rpy="0 0 0"/>
    <geometry>
      <box size="0.05 0.05 0.15"/>
    </geometry>
    <material name="red"/>
  </visual>
</link>
```

```
<?xml version='1.0'?>
<robot name="dd_robot">

  <!-- Base Link -->
  <link name="base_link">
    <visual>
      <origin xyz="0 0 0" rpy="0 0 0" />
      <geometry>
          <box size="0.5 0.5 0.25"/>
      </geometry>
    </visual>
  </link>

</robot>
```

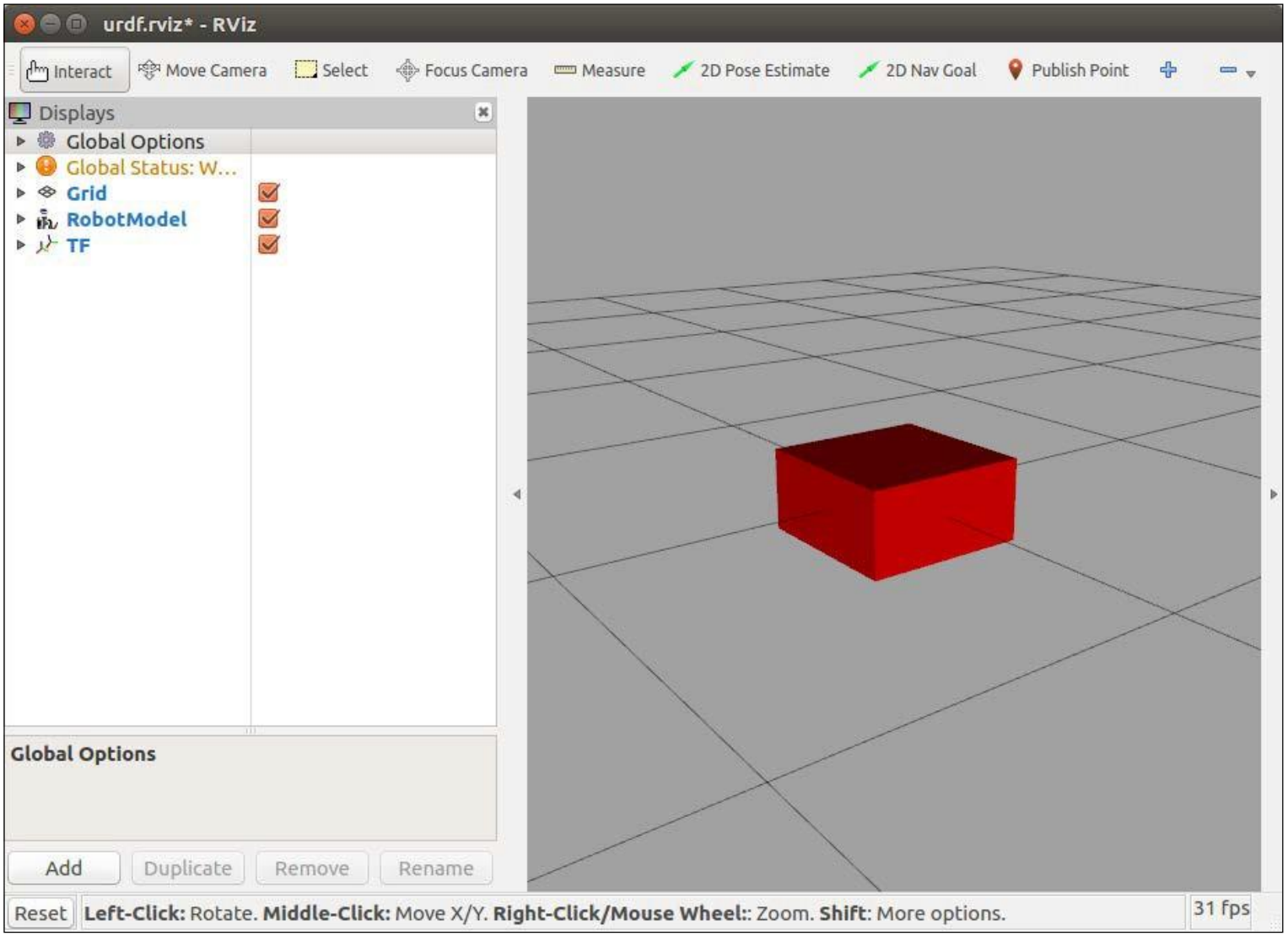
LAUNCH RVIZ AND THE ROBOT MODEL

```
$ roslaunch ros_robotics
ddrobot_rviz.launch model:=dd_robot.urdf
```

```
<launch>
  <!-- values passed by command line input -->
  <arg name="model" />
  <arg name="gui" default="False" />

  <!-- set these parameters on Parameter Server -->
  <param name="robot_description" textfile="$(find
ros_robotics)/urdf/$(arg model)" />
  <param name="use_gui" value="$(arg gui)"/>
```

```
 <!-- Start 3 nodes: joint_state_publisher,
robot_state_publisher and rviz -->
   <node name="joint_state_publisher"
pkg="joint_state_publisher" type="joint_state_publisher" />

   <node name="robot_state_publisher"
pkg="robot_state_publisher" type="state_publisher" />

   <node name="rviz" pkg="rviz" type="rviz" args="-d $(find
ros_robotics)/urdf.rviz" required="true" />
   <!-- (required = "true") if rviz dies, entire roslaunch will be
killed -->
</launch>
```

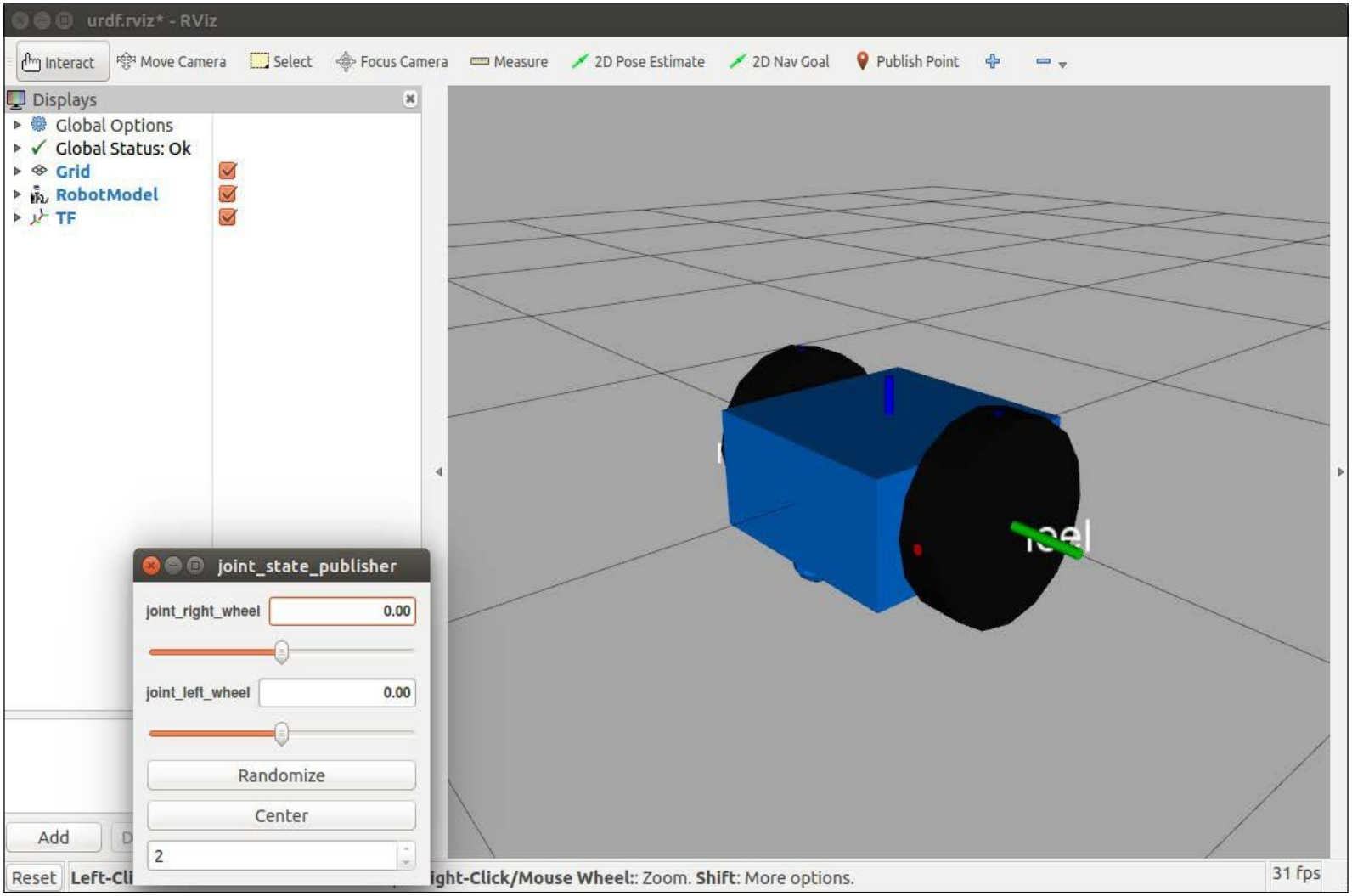
SUCCESS


urdf.rviz* - RViz
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Global Status: W...
Grid
RobotModel
TF
Global Options
Add
Duplicate
Remove
Rename
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel:: Zoom. Shift: More options.
31 fps

harman@harman-VirtualBox:~/catkin_ws/src/ros_robotics/urdf$ ls -la

```
total 36
drwxrwxr-x 2 harman harman 4096 Sep 24 18:47 .
drwxrwxr-x 3 harman harman 4096 Sep 24 18:44 ..
-rw-rw-r-- 1 harman harman 1083 Feb 21  2018 dd_robot2.urdf
-rw-rw-r-- 1 harman harman 1265 Feb 21  2018 dd_robot3.urdf
-rw-rw-r-- 1 harman harman 1466 Feb 21  2018 dd_robot4.urdf
-rw-rw-r-- 1 harman harman 2273 Feb 21  2018 dd_robot5.urdf
-rw-rw-r-- 1 harman harman 2955 Feb 21  2018 dd_robot6.urdf
-rw-rw-r-- 1 harman harman 2993 Feb 21  2018 dd_robot.gazebo
-rw-rw-r-- 1 harman harman  254 Feb 21  2018 dd_robot.urdf
```

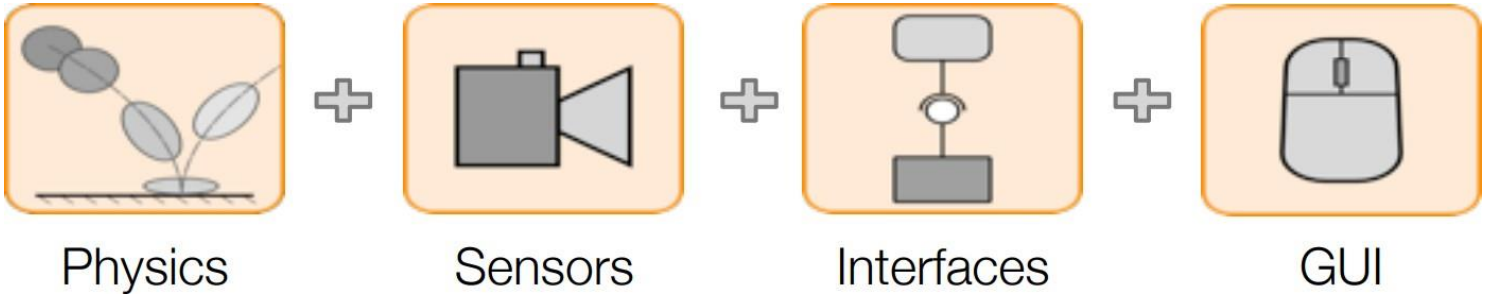
```
$ roslaunch ros_robotics ddrobot_rviz.launch
model:=dd_robot5.urdf gui:=True
```



dd_robot5.urdf in rviz

Gazebo simulator

- Goal: Best possible substitute for physical robot
- Architecture



- Advantages
  - Design and testing of robot's components and control
  - Software testing and verification (controllers)
  - Save time and money
- Installation : Built-in along with the ROS desktop-full.

Testing Gazebo

• Gazebo runs two executables: Gazebo server (simulation process) and Gazebo client (Gazebo GUI)

```
gazebo
```

• Add a square block and a sphere using the upper tool bar

• Right click on the sphere

• Select Apply Force/Torque

• Choose a value for the torque and force and select apply.

• Observe kinematics and dynamics simulation

Understanding ROS-Gazebo Filesystem

Controllers
.yaml files

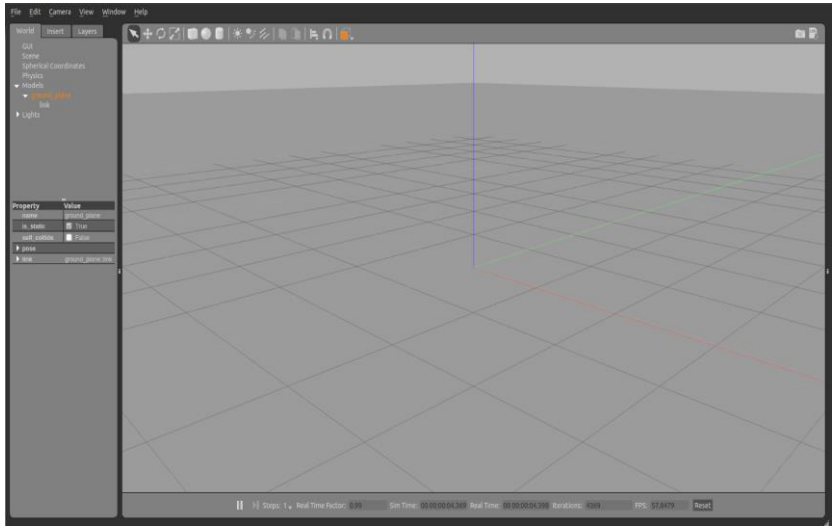
Robot modelling
.urdf, .xacro,
.gazebo files

Simulation
Environment
.world files

Notice that each one of this folders is a different package and usually they are located inside the robot's name folder. Packages' names are recommended but not mandatory.

World

```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <!-- Global light source -->
    <include>
      <uri>model://sun</uri>
    </include>
  </world>
</sdf>
```



```
<?xml version="1.0" ?>
<sdf version="1.4">
  <world name="default">
    <include>
      <uri>model://ground_plane</uri>
    </include>
    <!-- Global light source -->
    <include>
      <uri>model://sun</uri>
    </include>
    <!-- Include model of gas station-->
    <include>
      <uri>model://gas_station</uri>
      <name>gas_station</name>
      <pose>-2.0 7.0 0 0 0 0</pose>
    </include>
  </world>
</sdf>
```

THANKS TO

Guillermo Castillo
Department of Electrical and Computer Engineering Ohio State University