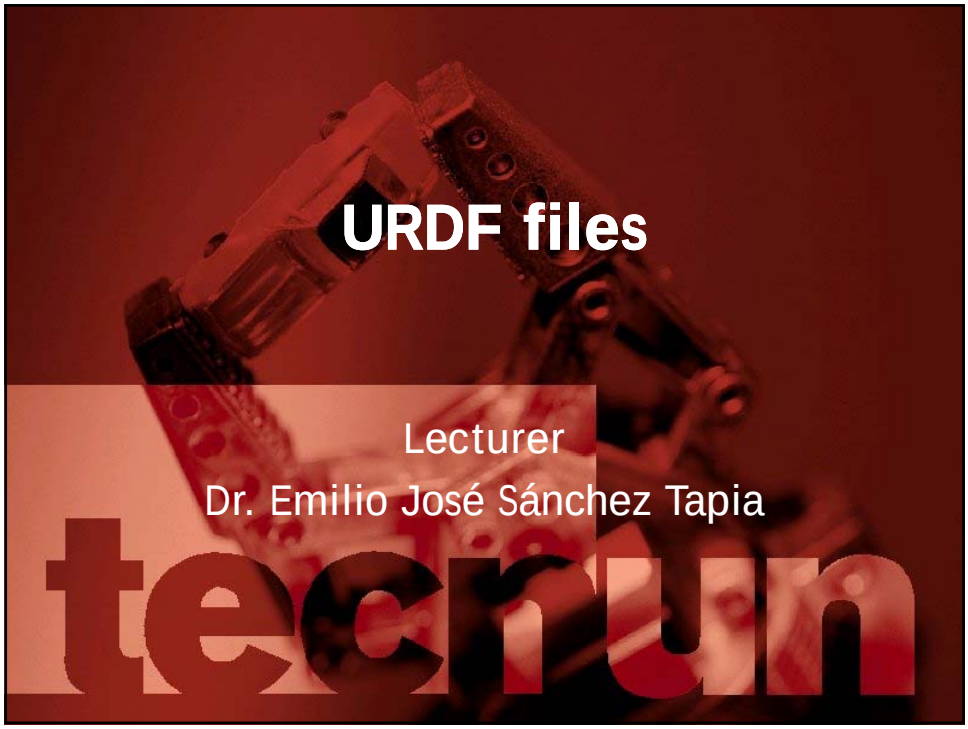
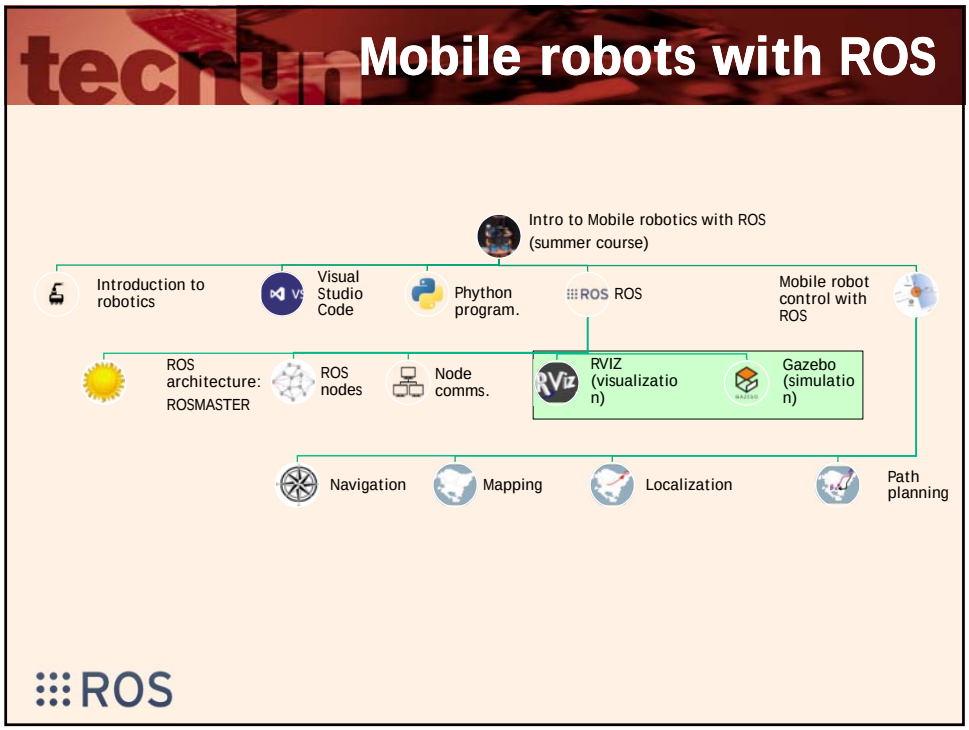
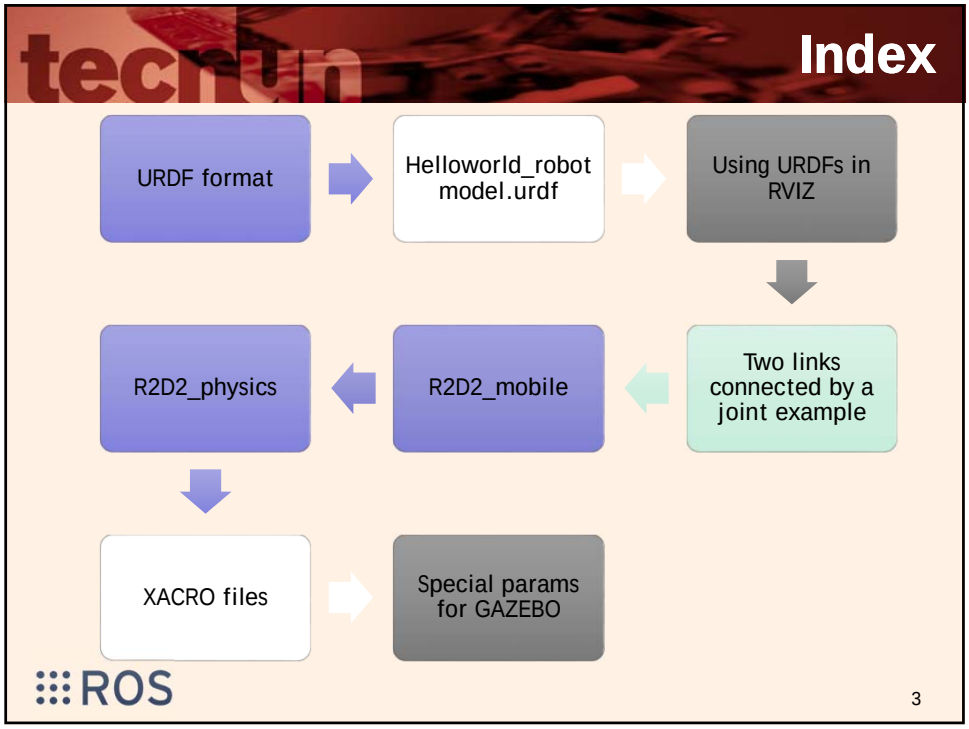
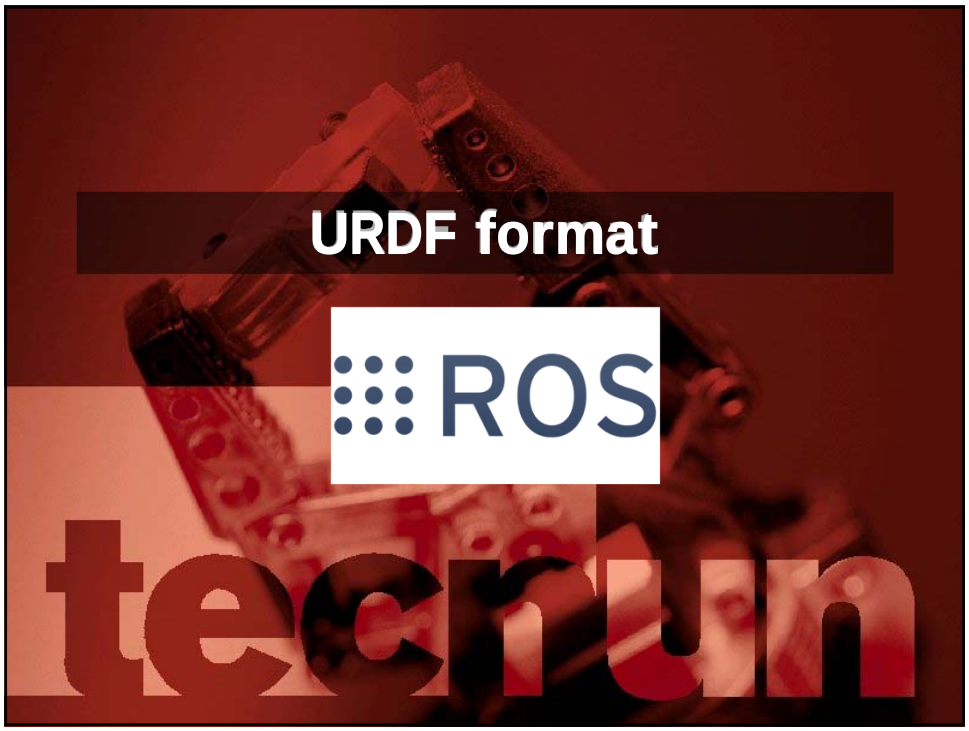

URDF files
Lecturer
Dr. Emilio José Sánchez Tapia
tecnun


tecnun
Mobile robots with ROS
Intro to Mobile robotics with ROS (summer course)
Introduction to robotics
Visual Studio Code
Phython program.
ROS
Mobile robot control with ROS
ROS architecture: ROSMASTER
ROS nodes
Node comms.
RVIZ (visualizatio n)
Gazebo (simulatio n)
Navigation
Mapping
Localization
Path planning
ROS


Index
URDF format
Helloworld_robot model.urdf
Using URDFs in RVIZ
Two links connected by a joint example
R2D2_mobile
R2D2_physics
XACRO files
Special params for GAZEBO
ROS
3


URDF format
ROS

Unified Robot Description Format (URDF)

- XML file that describes the robot model:
  - Kinematic and dynamic model
  - Robot visual representation
  - Collision model
- An URDF file can be generated from xacro files
- The robot description (URDF) is stored on the parameter server under `/robot_description`
- For Rviz (ROS visualizer) and GAZEBO (ROS simulator)
- To see a urdf file graphically, use *urdf_to_graphiz*
- http://wiki.ros.org/urdf/Tutorials

ROS

5

URDF: file format description

- A set of link elements and a set of joint elements
- Joints connect the links
- robot_model.urdf

```
<robot name="robot">
   <link> ... </link>
   <link> ... </link>
   <link> ... </link>
   <joint> .... </joint>
   <joint> .... </joint>
   <joint> .... </joint>
</robot>
```

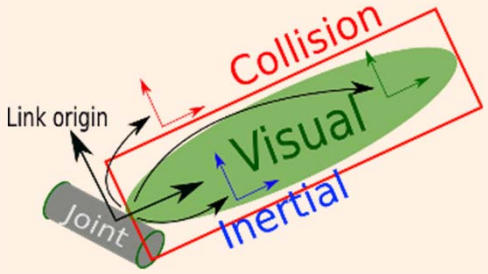
ROS

6

URDF: link description

Check http://wiki.ros.org/urdf/XML/link

- Inertial: origin (pose→[xyz and rpy] of the center of gravity, relative to the link, mass, inertia (Ixx, Ixy, Ixz, Izz, Izz)
- Visual: name,origin (pose→[xyz and rpy] with respect to the reference frame of the link), geometry (box, cylinder, sphere, mesh [*.dae (with colors) or *.stl (without colors)]), material> (name, color [rgba], texture[file])
- Collision: name, origin, geometry



ROS

7

URDF: link description

```
<link name="link_name">
   <visual>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
         <mesh filename="mesh.dae"/> <!-- with colors -->
      </geometry>
      <material name="Cyan">
       <color rgba="0 1.0 1.0 1.0"/>
      </material>
   </visual>
   <collision>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <geometry>
         <cylinder length="0.4" radius="0.25"/>
      </geometry>
   </collision>
   <inertial>
      <origin xyz="0 0 0" rpy="0 0 0"/>
      <mass value="10"/>
      <inertia ixx="10"  ixy="0"  ixz="0" iyy="10" iyz="0" izz="10" />
</inertial>
</link>
```

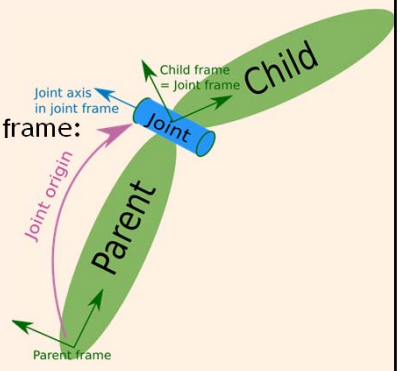
ROS

8

URDF: joint description

Check http://wiki.ros.org/urdf/XML/joint

- Type: revolute, continuous, prismatic, fixed, floating, and planar
- Parent and Child: to define to which links it is connected
- Origin: the xyz coord and rpy are referenced to the Parent axis, not the child axis.
- Limit: lower, upper, effort, velocity
- Dynamics: damping, friction
- Axis: the joint axis, specified in the joint frame:
  - Revolute: the axis of rotation
  - Prismatic: the axis of translation
  - Planar: the surface normal
  - Fixed and floating: do not use this option
- Safety_controller:
  - limits the joint boundaries
  - saturates the control action



ROS

9

URDF: joint description

```
<joint name="joint_name" type="revolute">
   <axis xyz="0 0 1"/>
   <limit effort="1000.0" upper="0.548" … />
   <origin rpy="0 0 0" xyz="0.2 0.01 0"/>
   <parent link="parent_link_name"/>
   <child link="child_link_name"/>
   <dynamics damping="0.0" friction="0.0"/>
   <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7"
/>
   <safety_controller k_velocity="10" k_position="15"
soft_lower_limit="-2.0" soft_upper_limit="0.5" />
</joint>
```

ROS

10

URDF: helloworld_robotmodel.urdf

- Create helloworld_robotmodel.urdf
- Save it in the urdf folder of the helloworld_pkg package
- The file:

```
<?xml version="1.0"?>
<robot name="myfirstRobot">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>
</robot>
```

ROS

12

URDF: helloworld_robotmodel.launch

- Create helloworld_robotmodel.launch
- Save it with other launch files
- The file:

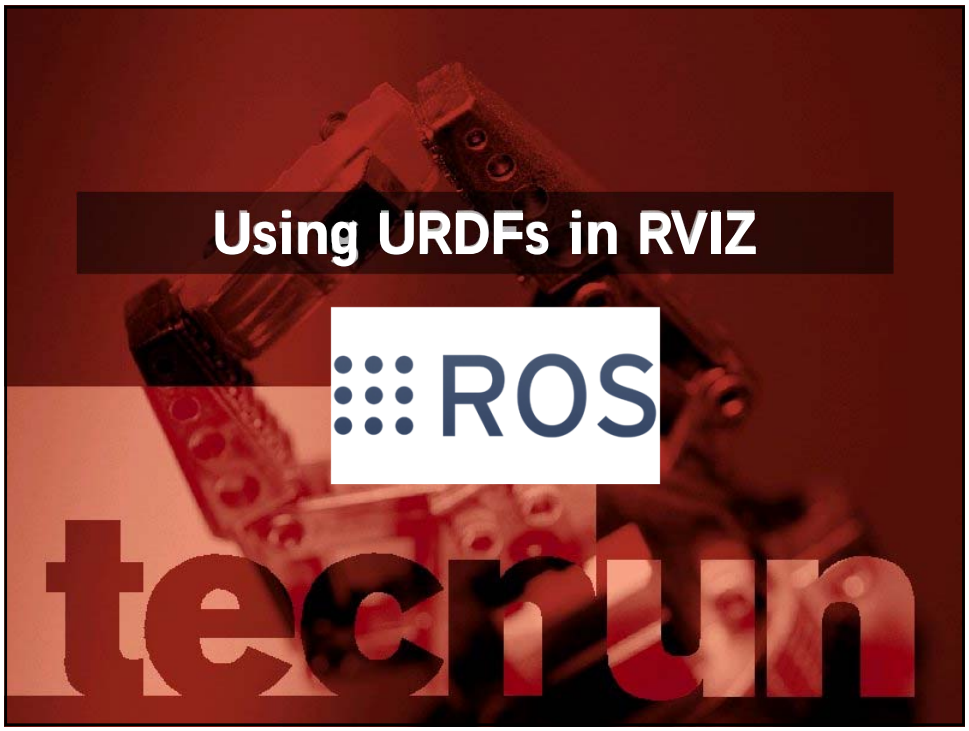
```
<launch>
  <param name="robot_description" textfile="$(find
helloworld_pkg)/urdf/helloworld_robotmodel.urdf" />
  <arg name="gui" default="False" />
  <param name="use_gui" value="$(arg gui)"/>
  <node pkg="tf" type="static_transform_publisher" name= "world_frame"
args="0 0 0 0 0 0 map base_link 100"/>
  <node name="rviz" pkg="rviz" type="rviz" />
</launch>
```

- Execute the launch file:

```
> roslaunch helloworld_pkg helloworld_robotmodel.launch gui:=true
```

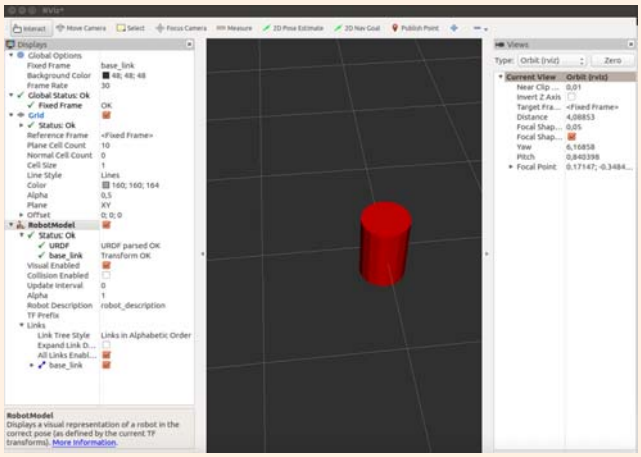
ROS

13



URDF: rviz configuration to see the robot model

- Select the base_link as a fixed frame
- Load the plugin robot model

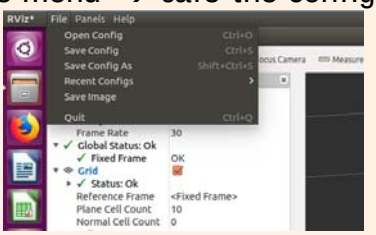


ROS

15

URDF: Save the configuration

- In the file menu → safe the config



- Modify the launch file changing the rviz node
  - Old line

```
<node name="rviz" pkg="rviz" type="rviz" />
```

  - Replace with

```
<node name="rviz" pkg="rviz" type="rviz" args="-d $(find
helloworld_pkg)/rviz/helloworld.rviz"/>
```

ROS

16

URDF: launch file template for RVIZ

```
<launch>
<!-- USE: roslaunch my_package urdf_visualize.launch model' -->
  <arg name="model" default=""/>
  <param name="robot_description" command="$(find myrobot_package)
                                           /urdf/myrobot.urdf" />
   <!- if we want a second (clone robot) -->
   <param name="robot2_description" command="$(find myrobot_package)
                                           /urdf/myrobot.urdf" />
   <!- if you do not have previously defined, include a TF -->
   <node pkg="tf" type="static_transform_publisher" name=
                "world_frame" args="0 0 0 0 0 0 map base_link 100"/>
   <!- add all nodes you need -->
   <node name="node name" pkg="package name" type="file">
    <param name="param name" value="param value"/> </node>
    <!-- Show in Rviz   -->
    <node name="rviz" pkg="rviz" type="rviz" args="-d
                $(find my_package)/rviz_config/urdf.rviz"/>-->
</launch>
```

ROS

17

Two links connected by a joint example

ROS

tecnun

URDF: example of two links connected by a joint

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
    </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
  </joint>
</robot>
```

ROS

19

URDF: example of two links connected by a joint

- Modify the launch file changing the TF node
  - Old line

```
<node pkg="tf" type="static_transform_publisher" name= "world_frame"
args="0 0 0 0 0 0 map base_link 100"/>
```

  - Replace with

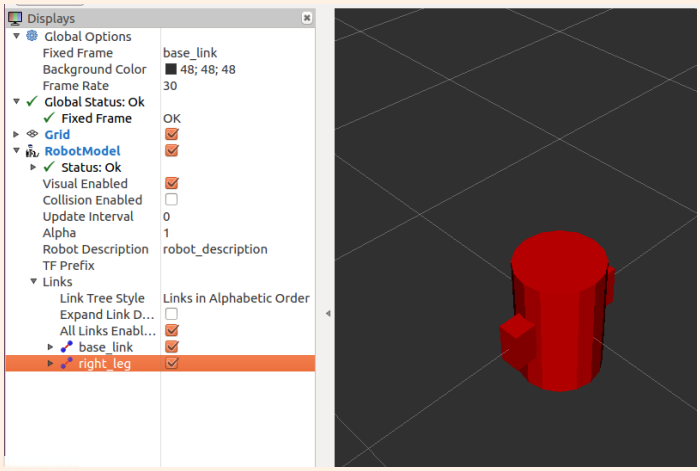
```
<node name="joint_state_publisher" pkg="joint_state_publisher"
type="joint_state_publisher" />
<node name="robot_state_publisher" pkg="robot_state_publisher"
type="state_publisher" />
```

ROS

20

URDF: example of two links connected by a joint

- Now, the launch file will activate a node that will create all TF we need for all links/joint we have in our model



ROS

21

URDF: simple 2 joints- adding origings

```
<?xml version="1.0"?>
<robot name="multipleshapes">
  <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
    </visual>
  </link>

  <link name="right_leg">
    <visual>
      <geometry>
        <box size="0.6 0.1 0.2"/>
      </geometry>
        <origin rpy="0 1.57075 0" xyz="0 0 -0.3"/>
      </visual>
  </link>

  <joint name="base_to_right_leg" type="fixed">
    <parent link="base_link"/>
    <child link="right_leg"/>
      <origin xyz="0 -0.22 0.25"/>
  </joint>

</robot>
```

ROS

22

URDF: simple 2 joints- adding material

```
<?xml version="1.0"?>
<robot name="multipleshapes">
    <material name="blue">
      <color rgba="0 0 0.8 1"/>
    </material>
    <material name="white">
      <color rgba="1 1 1 1"/>
    </material>

   <link name="base_link">
    <visual>
      <geometry>
        <cylinder length="0.6" radius="0.2"/>
      </geometry>
      <material name="blue"/>
    </visual>
  </link>

…

</robot>
```

ROS

23

URDF: R2D2_mobile

- Create the following launch file (r2d2_mobile.launch)

```
<launch>
  <param name="robot_description" textfile="$(find
helloworld_pkg)/urdf/r2d2_mobile.urdf" />

  <arg name="gui" default="False" />
  <param name="use_gui" value="$(arg gui)"/>

  <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
helloworld_pkg)/rviz/helloworld.rviz"/>
</launch>
```

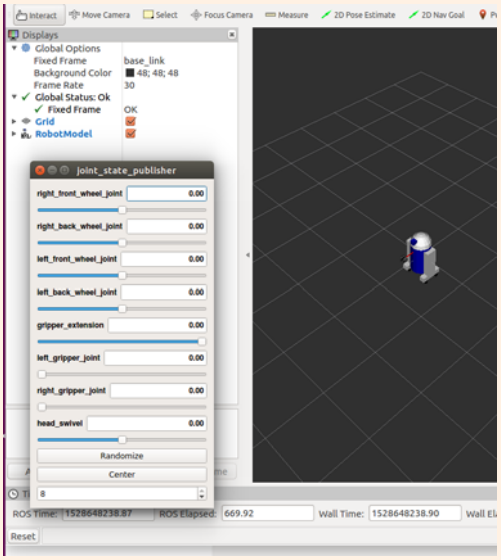
ROS

25

URDF: R2D2_mobile

- Copy the file R2D2_mobile.urdf in the urdf file of the helloworld_pkg
- Run the launch file

```
> roslaunch helloworld_pkg
r2d2_mobile.launch gui:=true
```



ROS

26

URDF: R2D2_physics

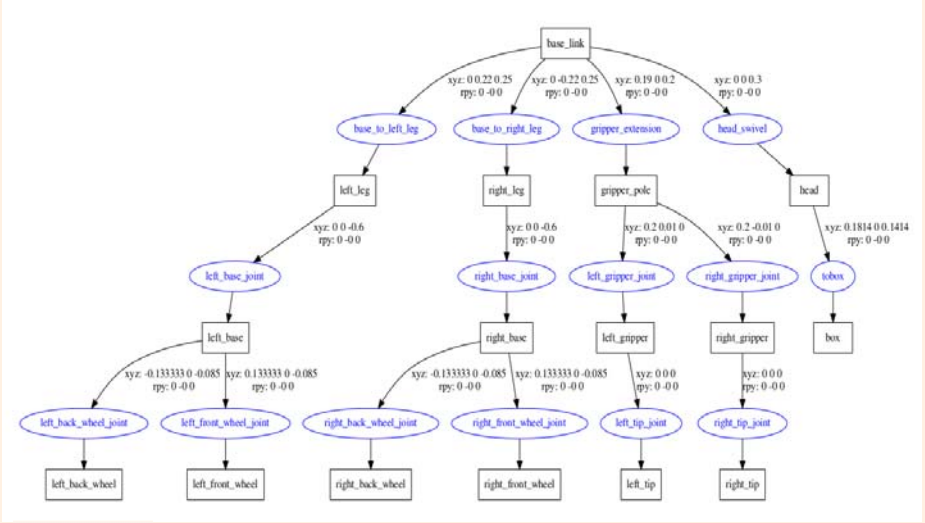
- View open R2D2_mobile.urdf with gedit or notepad or notepad++ and identify all html tags corresponding to the link and joint definition
- Now→ physics and collition data
  - Check the file : R2D2_physics.urdf
  - Try it with `rviz`
- Use the urdf_to_graphiz tool

```
$ urdf_to_graphiz r2d2_physics.urdf
Created file physics.gv
Created file physics.pdf
```

- Open and see the generated pdf

ROS 28

URDF: R2D2_physics graph



ROS

29

Xacro files

ROS

xacro files

- The building of a URDF file involved lots of math calculations
- Instead of using URDF format→ use XACRO files
- In XACRO files, we can use:
  - Constants
  - Simple Math
  - Macros
- XACRO→ a macro language to create URDF files
- Working procedure:
  - Create a XACRO file
  - Convert the XACRO file to URDF

```
rosrun xacro xacro model.xacro > model.urdf
```

  - Use the automatically generated URDF

ROS

31

XACRO files: launch file

- We can write the XACRO files and automatically translate them to URDF files → launch files
- Automatically generation of URDF files in a launch file → include the following line

```
<param name="robot_description" command="$(find
xacro)/xacro      '$(find
my_package)/urdf/my_robot_description.xacro'" />
```

- Compare to the old style if we use URDF directly

```
<param name="robot_description" textfile="$(find
my_package)/urdf/my_robot_description.urdf" />
```

ROS

32

XACRO files: constants

XACRO file → URDF file

```
<xacro:property name="width" value="0.2" />
<xacro:property name="bodylen" value="0.6" />
<link name="base_link">
  <visual>
    <geometry>
      <cylinder radius="${width}"
length="${bodylen}"/>
    </geometry>
    <material name="blue"/>
  </visual>
  <collision>
    <geometry>
      <cylinder radius="${width}"
length="${bodylen}"/>
    </geometry>
  </collision>
</link>
```

```
<link name="base_link">
  <visual>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
    <material name="blue"/>
  </visual>
  <collision>
    <geometry>
      <cylinder length="0.6" radius="0.2"/>
    </geometry>
  </collision>
</link>
```

ROS

33

XACRO files: MATH

XACRO file → URDF file

```
<link name="${5/6}"/>
```

```
<link name="0.833333333333"/>
```

- We can do simple maths using constants

```
<cylinder radius="${wheeldiam/2}" length="0.1"/>
<origin xyz="${reflect*(width+.02)} 0 0.25" />
```

ROS

34

XACRO files: PARAMETRIZED MACRO

XACRO file

```
<xacro:macro name="default_inertial"
params="mass">
    <inertial>
        <mass value="${mass}" />
        <inertia ixx="1.0" ixy="0.0" ixz="0.0"
            iyy="1.0" iyz="0.0"
            izz="1.0" />
    </inertial>
  </xacro:macro>
<xacro:default_inertial mass="10"/>
```

```
<origin rpy="4 5 6" xyz="1 2 3"/>
```

- BE CAREFUL: If the xacro with a specified name is not found, it will not be expanded and will NOT generate an error!
- In this example the paremeter is a simple number

ROS

35

XACRO files: PARAMETRIZED MACRO

XACRO file

- But the paremeter can be block of code!

```
<xacro:macro name="blue_shape"
params="name *shape">
  <link name="${name}">
    <visual>
      <geometry>
        <xacro:insert_block name="shape" />
      </geometry>
      <material name="blue"/>
    </visual>
    <collision>
      <geometry>
        <xacro:insert_block name="shape" />
      </geometry>
    </collision>
  </link>
</xacro:macro>
<xacro:blue_shape name="base_link">
  <cylinder radius=".42" length=".01" />
</xacro:blue_shape>
```

ROS

36

XACRO files: r2d2_macroed.urdf.xacro

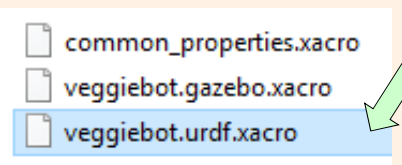
- Check the file 04.r2d2_macroed.urdf.xacro
- Create a launch file to see it in rviz
- Execute the lauch file

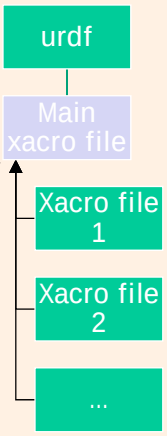
ROS

37

XACRO files: Veggiebot

- Xacro files are distributed in three files that combined → urdf
  - Common properties
    - Material properties
  - Gazebo
    - GAZEBO specific properties (contact models, gazebo plugins)
  - URDF
    - Main file includes the other two
    - With visual, collision and inertial fiel





ROS

38

XACRO/URDF files: special tags for GAZEBO

- In a URDF/XACRO files we can use tags only understood by GAZEBO.
- These tags controls the properties of the physis engine and GAZEBO plubings.
- This must to be studied during the GAZEBO lecture.

ROS

40

URDF:link contact coefficients

- To define how the links behave when they are in contact with one another.
- Typical scenario: wheels contacting the floor
- Three types
  - mu - The friction coefficient
  - kp - Stiffness coefficient
  - kd - Dampening coefficient

ROS

41