Introduction to Robot Modeling in ROS

Understanding URDF and XACRO

Guillaume Walck

November, 2015

Table of Content

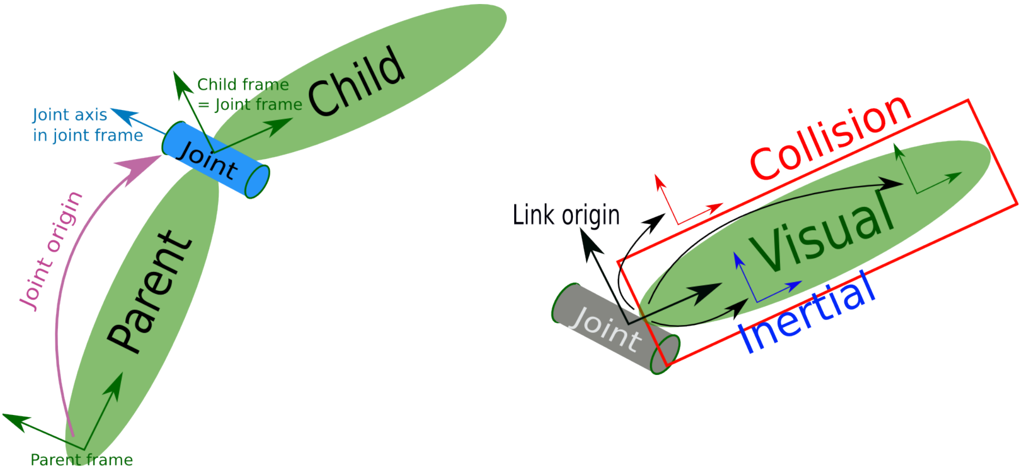
Outline

1 URDF
- Concepts
- Basic usage
- Advanced usage
- Use-case

URDF concepts

- What:
  - Unified Robot Description Format
  - Kinematic and basic physics description of a robot
- How:
  - XML format
  - Tags: link, joint, transmission, ...
  - Kinematic tree structure
  - Order in the file does not matter

Link and joint representation



Link element (1)

Robot link with one frame of reference

- Syntax:
  - name
- child element *visual*
  - visual description of the link
  - can be multiple (union of all)
  - geometry primitives (box, cylinder, sphere)
  - geometry meshes (resources stl/dae)
  - origin: placement relatively to link reference frame (rpy = fixed axis rotation)
  - material

example

```
<link name="forearm">
 <visual>
  <geometry>
   <origin xyz="0 0 0.1" rpy="0 0 0" />
   <box size="0.1 .2 .5"/>
  </geometry>
  <material name="Cyan">
   <color rgba="0 1.0 1.0 1.0"/>
  </material>
 </visual>
</link>
```

Link element (2)

Robot link with one frame of reference

- Syntax:
    - name
- child element *visual*
    - visual description of the link
    - can be multiple (union of all)
    - geometry primitives (box, cylinder, sphere)
    - geometry meshes (resources stl/dae)
    - origin: placement relatively to link reference frame (rpy = fixed axis rotation)
    - material

example 2

```
<link name="gripper">
 <visual>
  <geometry>
   <mesh filename="package://pkg/m.dae"/>
  </geometry>
 </visual>
 <visual>
  <geometry>
   <cylinder length="0.6" radius="0.2"/>
  </geometry>
 </visual>
</link>
```

Joint element

Robot joint between two links

- Syntax:
  - name
  - type: continuous , fixed, revolute, prismatic, planar, floating
- child element *parent*
- child element *child*
- child element *origin*
  - always in parent reference frame
- child element *axis*
  - for prismatic and revolute
  - in local joint reference frame

example

```
<joint name="joint1" type="revolute">
  <parent link="forearm"/>
  <child link="gripper"/>
 <origin xyz="0.5 0 0" rpy="0 0 -1.57" />
 <axis xyz="0 0 1" />
</joint>
```

Advanced link element (1)

Physics and collision description

- child element *collision*
  - similar to visual description of the link
  - can be multiple (union of all)
  - mesh resolution should be low

example

```
<collision>
 <geometry>
  <origin xyz="0 0 0.1" rpy="0 0 0"/>
  <mesh filename="package://pkg/x.dae"/>
 </geometry>
</collision>
```

Advanced link element (2)

Physics and collision description

- child element *inertial*
  - center of mass
  - mass
  - inertia matrix

example

```
<inertial>
 <origin xyz="0.5 0 0" rpy="0 -1.57 0"/>
 <mass value="10"/>
 <inertia ixx="0.4" ixy="0.0" ixz="0.0"
          iyy="0.4" iyz="0.0" izz="0.2"/>
</inertial>
```

Advanced joint element (1)

Physical limits, and dynamic properties

- child element *limit*
  - lower and upper rotation/translation limits
  - maximum velocity
  - maximum effort
- child element *dynamics*
  - friction
  - damping

example

```
<limit effort="1000.0"
      lower="0.0"
      upper="0.548"
      velocity="0.5" />

<dynamics damping="0.1" friction="0.1"/>
```

Advanced joint element (2)

Kinematic properties

- child element *mimic*
  - one joint follows another
  - $\mathit{value} = \mathit{multiplier} \times \mathit{other\_joint\_value} + \mathit{offset}$

example

```
<joint name="joint2" type="revolute">
  <mimic joint="joint1"
        multiplier="0.5"
        offset="0.1"/>
</joint>
```

Additional elements (1)

Transmission between joint and actuator

- element *transmission*
  - type
  - joint
  - actuator

example

```
<transmission name="j1_transmission">
 <type>sr_mechanism_model/Transmission</type>
 <actuator name="J1">
  <mechanicalReduction>1</mechanicalReduction>
 </actuator>
 <joint name="joint1">
  <hardwareInterface>EffortJointInterface
  </hardwareInterface>
 </joint>
</transmission>
```

Additional elements (2)

Gazebo setting

- element *gazebo*
  - reference
  - sensors
  - plugins
  - additional properties (self collide, gravity enable, ...)

example

```
<gazebo reference="forearm">
 <sensor type="contact" name="arm_cont">
  <contact>
   <collision>arm_collision</collision>
   <topic>arm_collision</topic>
  </contact>
<plugin name="b" filename="libgazebo_ros_bumper.so">
   <frameName>forearm</frameName>
   <bumperTopicName>/arm_col</bumperTopicName>
  </plugin>
 </sensor>
 <selfCollide>true</selfCollide>
</gazebo>
```

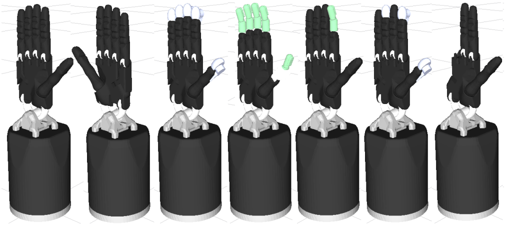
Simple use-case: Kuka arm

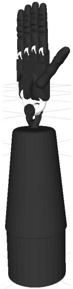
- Robot characteristics:
  - Serial manipulator
  - 7 DOF
- URDF:
  - One file
  - Easy 8 link + 7 joint description

⟹ little redundancy (4 different link shapes), can be read and and maintained, etc...

Advanced use-case: Shadow hand

- Robot characteristics:
  - Usually 5-fingered hand, 4 of which are identical
  - maximum of 24 DOF
  - Various fingertip & transmission models
  - Specific versions with less fingers





Advanced use-case: Shadow hand

- URDF:
  - One file per hand type, per transmission type and per fingertip model
  - Every link and joint is described explicitly

$\Longrightarrow$ a lot of redundancy, very long files, hard to read and hard to maintain, etc...

Outline

2 XACRO

- Concept
- Basic usage
- Use-case
- References

Concept

- What:
  - XML Macro language used for URDF simplification
  - Increase modularity
  - Reduce redundancy
  - Permit Parametrization
  - Generate URDF on-the-fly
- How:
  - Inclusion
  - Macros
  - Properties
  - Expansion of all xacro statements
  - Command line and output to stdout

Basic usage (1)

Every xml elements starts with *xacro*

- Properties:
  - definition
  - instantiation
  - string concatenation
- Simple math
  - in variables
  - nested variables
  - no function

example

```
<xacro:property name="width" value=".2"/>
<cylinder radius="${width}" length=".1"/>

<link name="${robotname}s_leg" />

<cylinder radius="${diam/2}" length=".1"/>
```

Basic usage (2)

- Simple macro:
  - definition
  - instantiation
- Parametrized macro:
  - definition
  - instantiation
- Nested macros

example

```
<xacro:macro name="default_origin">
 <origin xyz="0 0 0" rpy="0 0 0"/>
</xacro:macro>
<xacro:default_origin />
<xacro:macro name="default_inertial" params="mass">
 <inertial>
  <xacro:default_origin />
  <mass value="${mass}" />
   <inertia ixx="0.4" ixy="0.0" ixz="0.0"
          iyy="0.4" iyz="0.0" izz="0.2"/>
 </inertial>
</xacro:macro>
<xacro:default_inertial mass="10"/>
```

Basic usage (3)

- Default values:
  - Provides default values for optional or repeated parameters
- Conditional statement:
  - Only tests true or false 0 and 1
- Command line argument:
  - xacro.py file.xacro rad:=3

example

```
<xacro:macro name="pos" params="x y:=0"/>
<xacro:pos x="1"/>

<xacro:if value="<expression>">
<xacro:unless value="<expression>">

<xacro:arg name="rad" default="2"/>
<cylinder radius="$(arg rad)" length=".1"/>
```

Typical application

- Reduce redundant code
  - Repeated links should be defined as macros and called with parameters
  - Typical parameters: prefix, reflect
- Parametrized entities
  - Use parameters for length of links
  - Use math for origin or inertia calculation
  - Shape parameters according to length
- Modularity:
  - Generic code can be put as include, to be reused in other files
  - Separate concerns to easily deactivate parts of the urdf (remove gazebo tags)

Shadow hand with xacro

- Chosen solution:
  - One file per phalanx (link + joint assembly)
    with selectable transmission model and/or fingertip model
    (proximal / middle / distal / thproximal / thmiddle / thdistal)
  - One file per finger type (finger / thumb), including phalanges
  - One file per hand type including 5 or less fingers

References and documentation

- References:
  - ROS Wiki `wiki.ros.org/urdf`
  - Shadow Hand: `github.com/shadow_robot/sr_common/sr_description`
- Suggested documentation:
  - URDF Tutorial `wiki.ros.org/urdf/Tutorials/BuildingaVisualRobotModelwithURDFfromScratch`
  - Xacro Tutorial `wiki.ros.org/urdf/Tutorials/UsingXacrotoCleanUpaURDFFile`

Thank you ...

... for your attention!