KU LEUVEN GENT

Creating and Using an URDF File

Understanding Robot Description, Simulation, and Design

Luca Morello

Brief Summary

First Lesson

Kinematics - Summary

Definition	• The study of motion without considering forces. • Focuses on the relationship between robot's joint parameters and its end-effector position/orientation.
Types of Kinematics:	• **Forward Kinematics (FK)**: Position and orientation of the end-effector from joint angles. • **Inverse Kinematics (IK)**: Joint angles required to achieve a desired position and orientation from the end-effector position
Key Terms:	• **Degrees of Freedom (DOF)**: Number of independent movements a robot can make. • **Transformation Matrices**: Used to describe the position and orientation of the robot's linkages in space. • **Denavit-Hartenberg (DH) Parameters**: Standard method to represent the kinematic chain of robotic arms.

Dynamics - Summary

Definition	• study of forces and torques that cause motion in a robot. It involves modeling the physical interactions between the robot's joints, links, and environment.
Key Concepts	• Equations of Motion: Describes the dynamic behavior of the robot using Newton-Euler or Lagrangian methods.
Key Terms:	• Inertia Matrix: Describes the distribution of mass and its effect on the robot's motion. • Coriolis and Centrifugal Forces: Forces that arise in rotating systems, affecting the robot's movement. • Gravitational Forces: Forces acting on the robot due to gravity, influencing joint torques and motions.

Static Equilibrium - $\tau = J^T F$

What it means

- This equation connects the forces on a robot's end-effector to the required torques at the robot's joints.

Key Concepts:

- $\tau \rightarrow$ Torques needed at each joint to keep the robot in balance.
- $J^T \rightarrow$ Matrix that helps translate forces from the end-effector to the joints.
- $F \rightarrow$ Forces acting on the robot's end-effector (like holding an object or interacting with the environment).

Why It's Important:

- **Force Control: Helps control the forces the robot applies.**
- Static Analysis: Determines how much force each joint needs to balance the robot when external forces are applied.

Second Lesson

Robot Simulation

What it means

- script was used to simulate a robot, follow a trajectory, and calculate the necessary forces and energy.

Steps in the Script:

- Trajectory Following (Inverse Kinematics): The robot follows a desired path by calculating the joint angles needed to reach the end-effector's target position.
- Torque Calculation (Inverse Dynamics): Inverse Dynamics is used to compute the torques required at each joint to follow the trajectory.
- The torques are determined by considering the robot's mass, acceleration, and forces acting on the system.
- Power Calculation:The power required at each joint is calculated using the formula:

$$P = \tau \cdot \dot{q}$$

Introduction to Urdf

The Challenge of Robot Design and Testing

Scenario

- You are designing a robot and need to make important decisions about its components, like motors, joint parameters, and dynamics.

Challenges

- Prototyping: Building and testing physical prototypes is time-consuming and costly.
- Simplified Calculations: Basic calculations might only consider part of the system (like the payload), leading to inaccurate results.
- Complex Analytical Solutions: Trying to solve everything manually with traditional analytical methods can be very difficult and slow.

The Challenge of Robot Design and Testing

Why Use a Numerical Approach?

- Instead of relying on physical prototypes or complex manual calculations, a numerical solution can provide more accurate and faster results.
- **Rapid Prototyping:** Quickly iterate designs to resolve issues before physical implementation.
- **Risk-Free Testing:** Validate robot behaviors and control strategies without risking hardware.
- **Cost Reduction:** Identify potential problems early, saving on material and labor costs.

Question

- If your robot has a unique design that doesn't fit exactly into existing toolboxes, how can you calculate everything you need like motor requirements, forces, and torques?

Solution: Build a Digital Twin with a URDF File

What is a Digital Twin?

- A Digital Twin is a virtual model of your robot that allows you to simulate its behavior and test different configurations.

How to Build It

- URDF File: Create a detailed model of your robot using a URDF (Unified Robot Description Format) file.

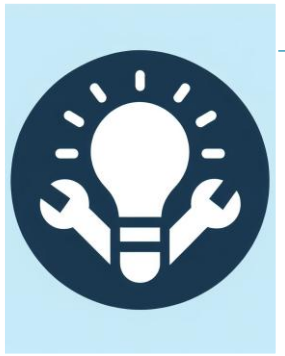
Why URDF?

- Advanced Functionality: With this model, you can use functions like inverse kinematics and inverse dynamics to calculate necessary parameters, such as torques, forces, and power requirements.
- Compatibility: URDF is widely supported across various systems and simulation tools (e.g., ROS, Gazebo, V-REP).
- Flexible: Unlike proprietary formats like rigidBodyTree in MATLAB, URDF is an open format that works seamlessly in multiple environments, making it easier to integrate with other tools and platforms.

Summary URDF

Definition: URDF stands for Unified Robot Description Format.

Purpose: Describes the structure of a robot including geometry, kinematics, and dynamics.:



Applications: Used in simulation tools like MATLAB, ROS, and Python frameworks.

Key note: Files .urdf are written in XML, it is possible to edit with any **Text Editors, ROS Editors**

Key Components of URDF (What Makes Up a URDF)

Kinematics: How Parts Are Connected and Move

- Kinematics just means how the parts of your robot are connected and how they can move.
- Links are the parts of the robot (like the arm or the base).
- Joints are what connect the parts together (like the shoulder joint that connects the arm to the body).
- Parent and Child:

Key Components of URDF (What Makes Up a URDF)

Dynamics: What the Robot is Made Of (Weight, Balance, etc.)

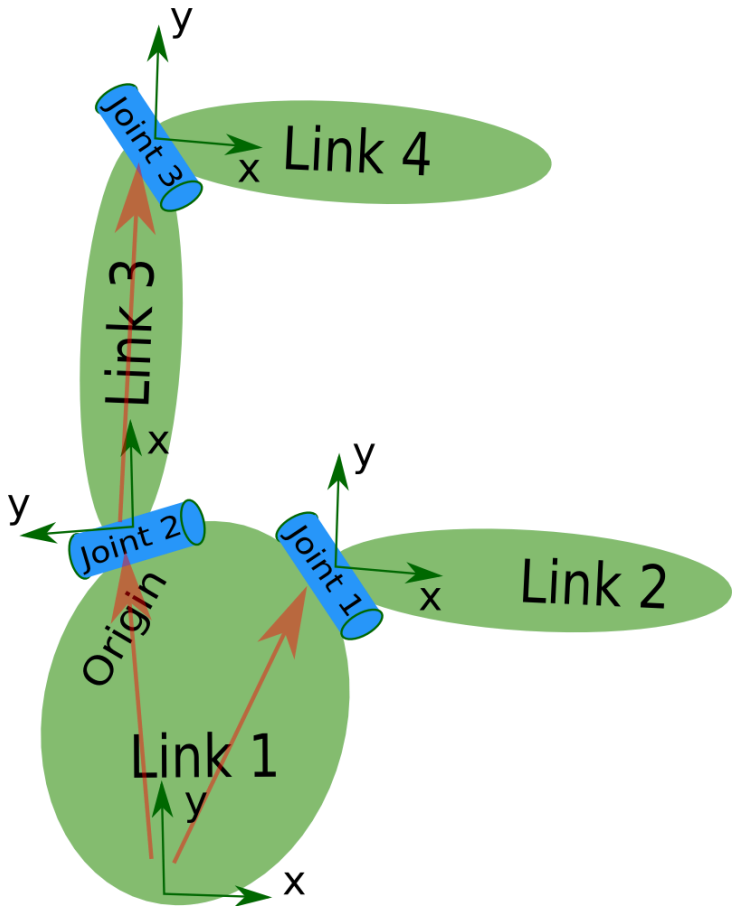
- Dynamics are about the robot's weight and how it behaves physically.
- Mass: This tells the computer how heavy each part of the robot is. For example, if the arm is made of metal, it will have a certain weight.
- Center of Mass: This is the balance point of the part. Imagine balancing a stick on your finger — the point where it balances is the center of mass.
- Inertia: This tells how much the part resists rotating or moving. For example, it's harder to spin a heavy object than a light one.

Key Components of URDF (What Makes Up a URDF)

Additional Elements: Extra Details to Help with Simulation

- Visualization: Seeing Your Robot Visualization helps you see what your robot looks like in the computer.
  - Think of it like creating a 3D model of the robot.
  - You can see what the robot's parts look like (boxes, cylinders, etc.).
- Collision: Making Sure the Robot Doesn't Go Through Things Collision is for checking if the robot runs into things while moving.
  - It's like making sure the robot doesn't walk through walls in a video game.
  - This helps make simulations realistic and avoids problems in real life.

Key Components of URDF (What Makes Up a URDF)



The description of a robot consists of:

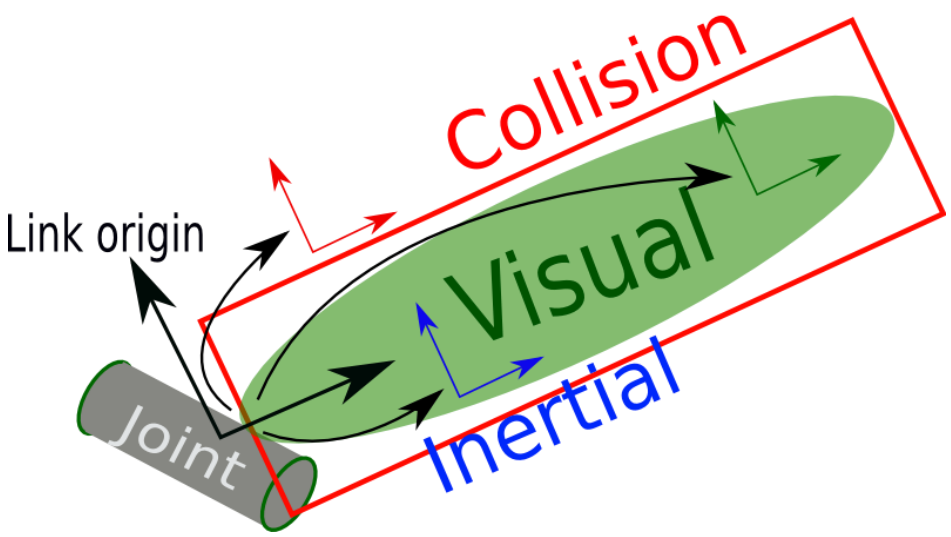
- A set of link elements
- A set of joint elements connecting the links together.

So, a typical robot description looks something like this:

```
<?xml version="1.0"?>
<robot name="pr2" >
    <link> ... </link>
    <link> ... </link>
    <link> ... </link>

    <joint>  ....  </joint>
    <joint>  ....  </joint>
    <joint>  ....  </joint>
</robot>
```

<link> element



The link element describes a rigid body with an inertia, visual features, and collision properties.

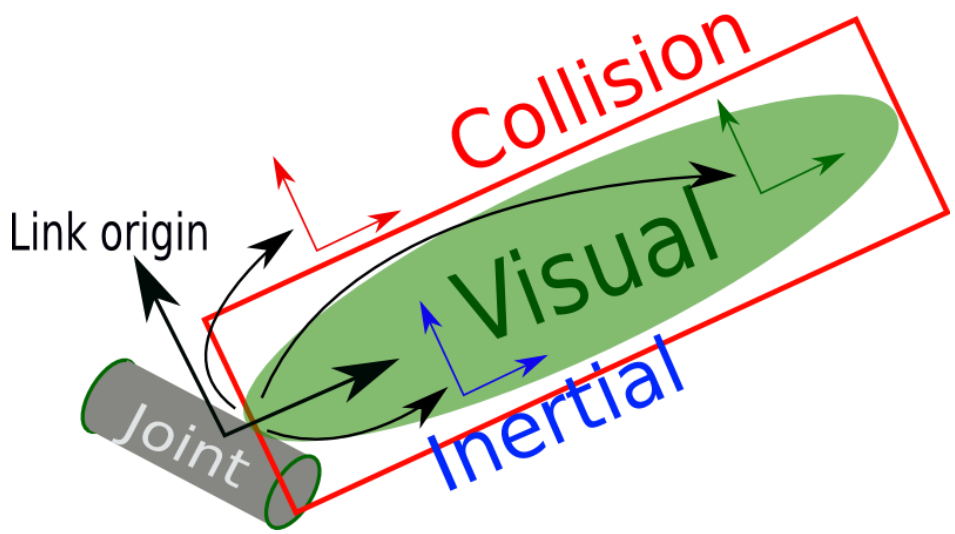
```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </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 radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

<link> element

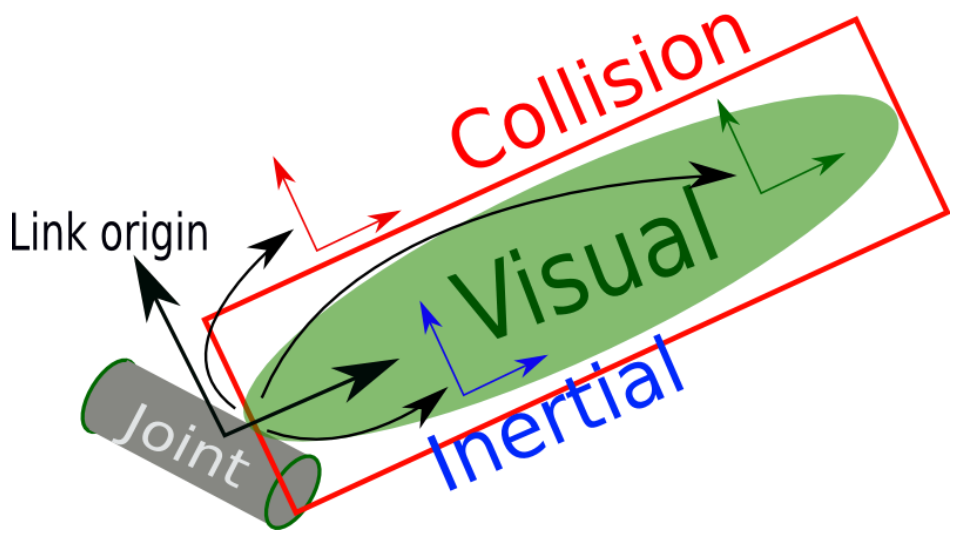
```
<inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
</inertial>
```



<inertial> *(Defaults to a zero mass and zero inertia if not specified)*
The link's mass, position of its center of mass, and its central inertia properties.

<link> element

```
<inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
</inertial>
```



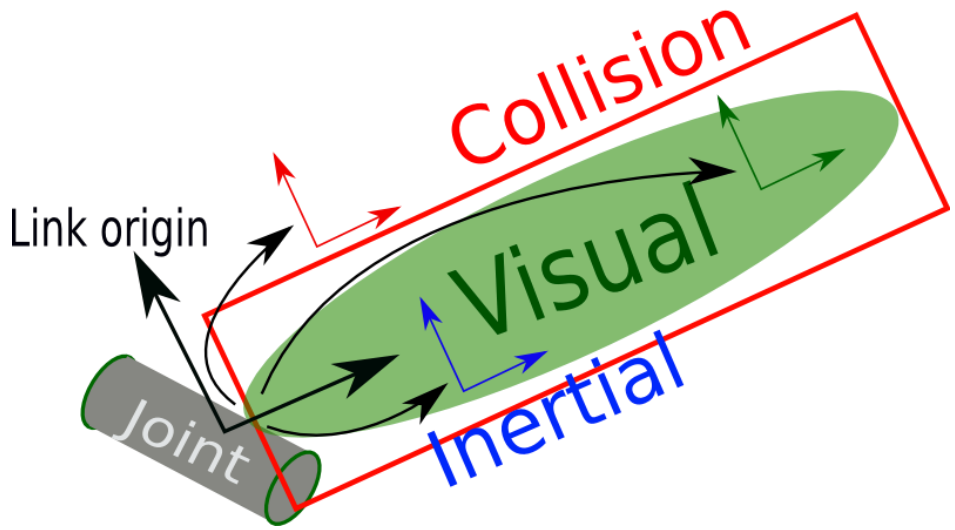
<origin>
This pose (translation, rotation) describes the position and orientation of the link's center of mass frame relative to the link-frame

xyz *(Defaults to zero vector)*
Position vector from link-frame origin to link's center of mass

rpy *(Defaults to identity if not specified)*
Represents the orientation of the center of mass frame as a sequence of Euler rotations (r p y) in radians.

<link> element

```
<inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100"  ixy="0"  ixz="0" iyy="100" iyz="0" izz="100" />
</inertial>
```

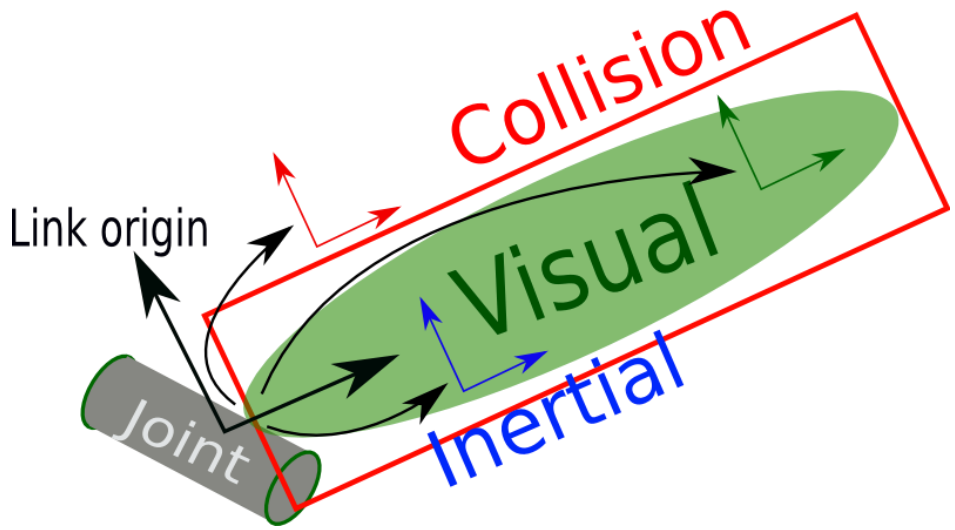


<mass>
The mass of the link is represented by the **value** attribute of this element

<inertia>
This link's moments of inertia **ixx, iyy, izz** and products of inertia **ixy, ixz, iyz** respect to the link's center of mass

<link> element

```
<visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
</visual>
```



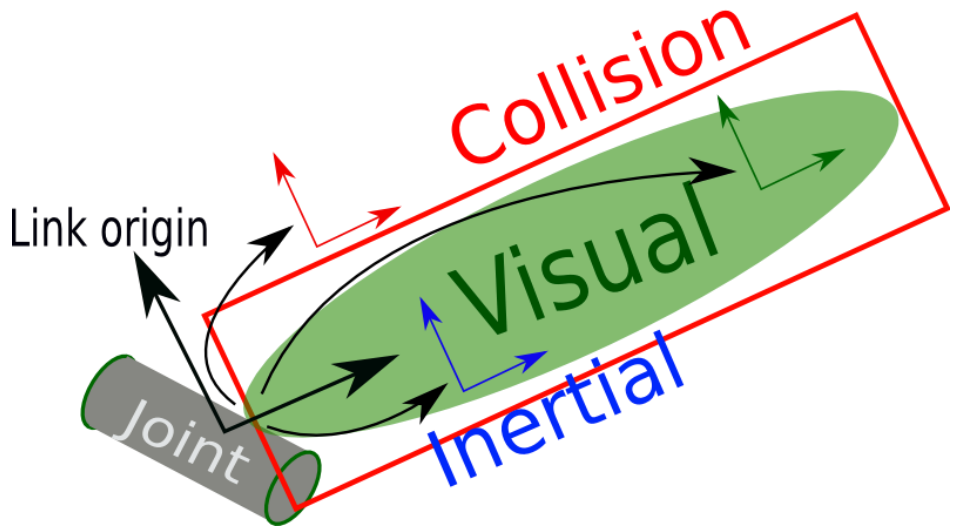
<visual> *(Optional)*
The visual properties of the link.

This element specifies the **shape of the object** (box, cylinder, etc.) **for visualization purposes**.

Note: multiple instances of <visual> tags **can** exist for the same link. The union of the geometry they define forms the visual representation of the link

<link> element

```
<visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
</visual>
```



<origin> *(Defaults to identity if not specified)*
The reference frame of the visual element with respect to the reference frame of the link.
xyz *(Defaults to zero vector if not specified)*

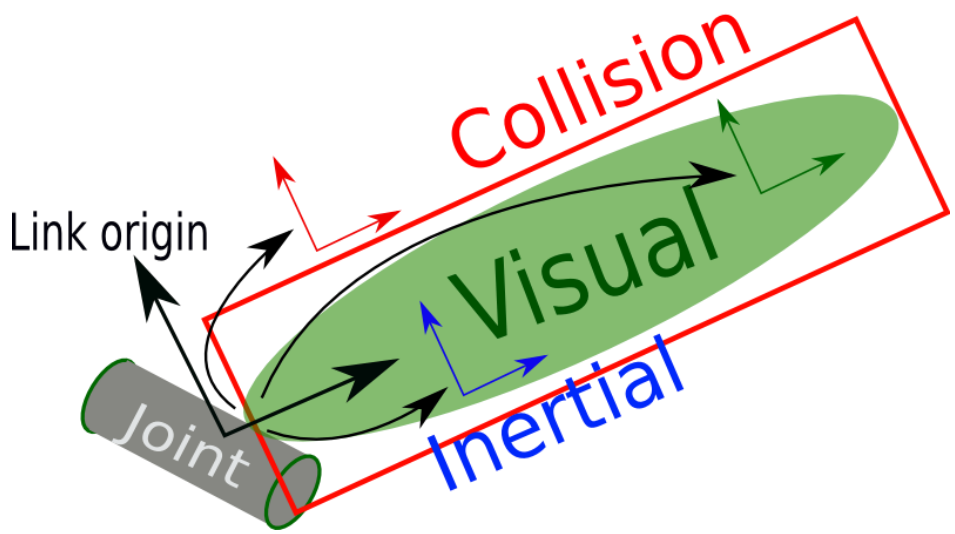
- Represents the **x**, **y**, **z** offset.

rpy *(Defaults to identity if not specified)*

- Represents the fixed axis roll, pitch and yaw angles in radians.

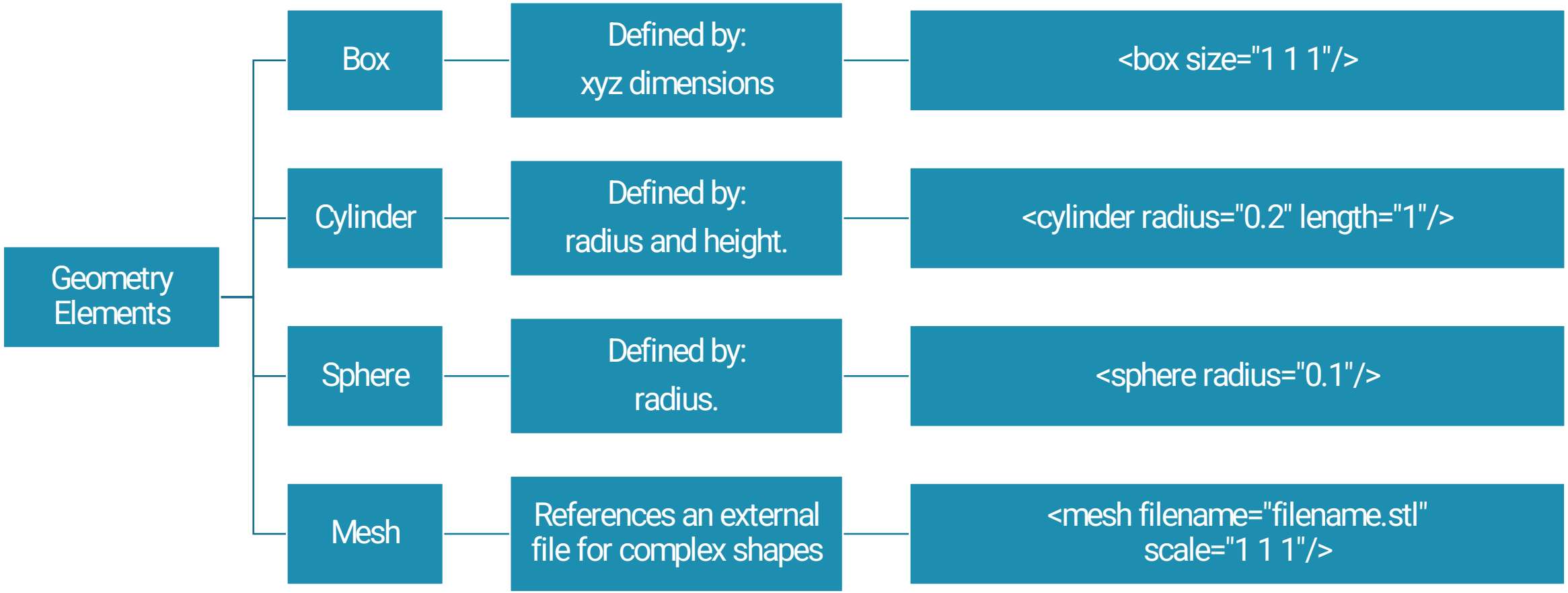
<link> element

```
<visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
</visual>
```



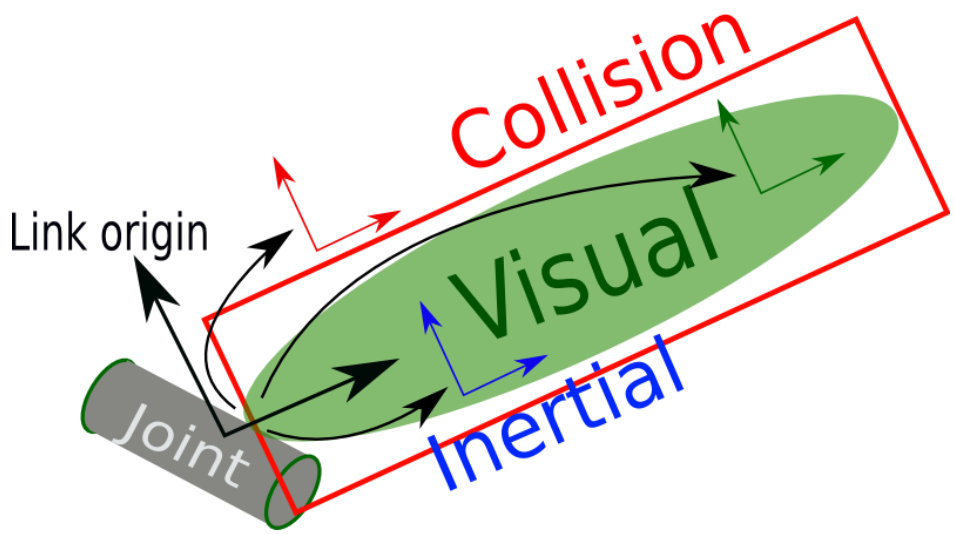
<geometry> *(required)*
The shape of the visual object. This can be *one* of the following:

URDF Example - Link



<link> element

```
<collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
</collision>
```



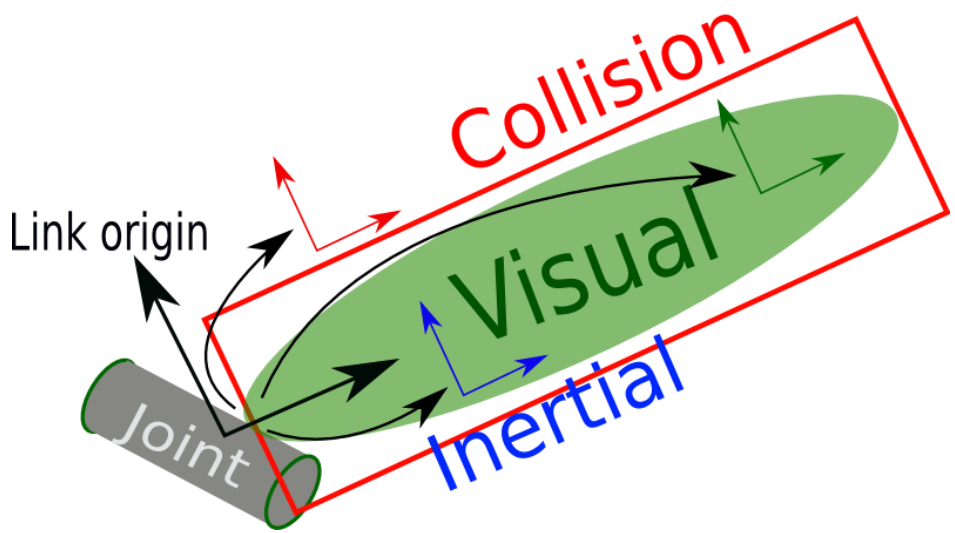
<collision> *(optional)*
The collision properties of a link.

Note that this **can be different from the visual properties of a link**, for example, simpler collision models are often used to reduce computation time.

Note: multiple instances of <collision> tags can exist for the same link. The union of the geometry they define forms the collision representation of the link.

<link> element

```
<collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
</collision>
```



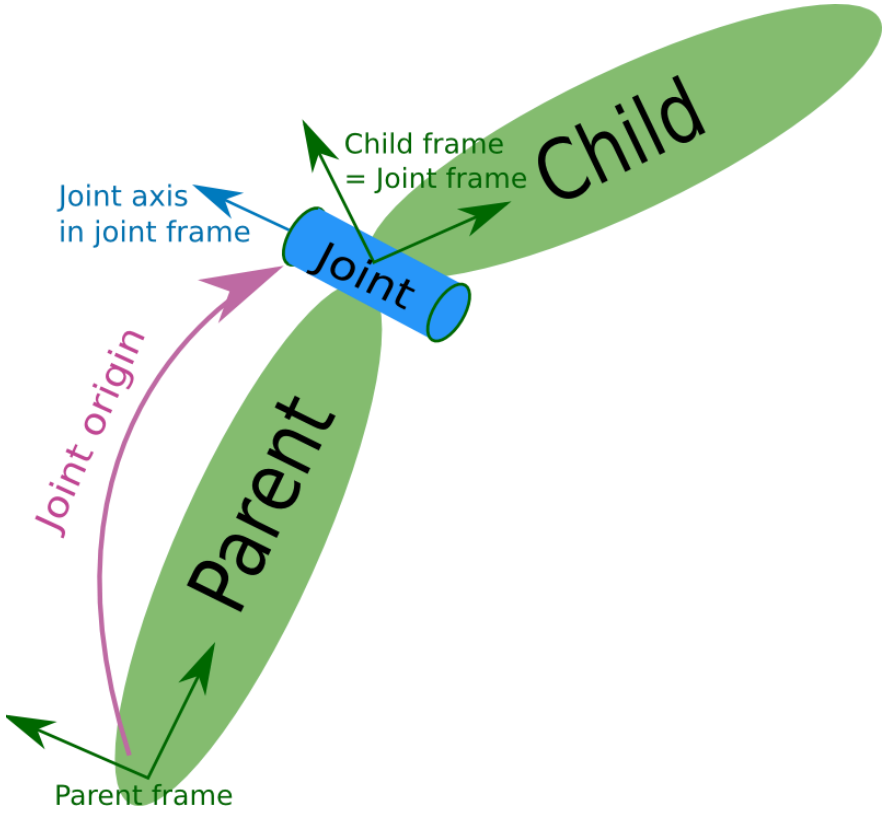
<origin> *(Defaults to identity if not specified)*
The reference frame of the collision element, relative to the reference frame of the link.
xyz *(Defaults to zero vector if not specified)*

- Represents the **x**, **y**, **z** offset.

rpy *(Defaults to identity if not specified)*

- Represents the fixed axis roll, pitch and yaw angles in radians.

<Joint> element



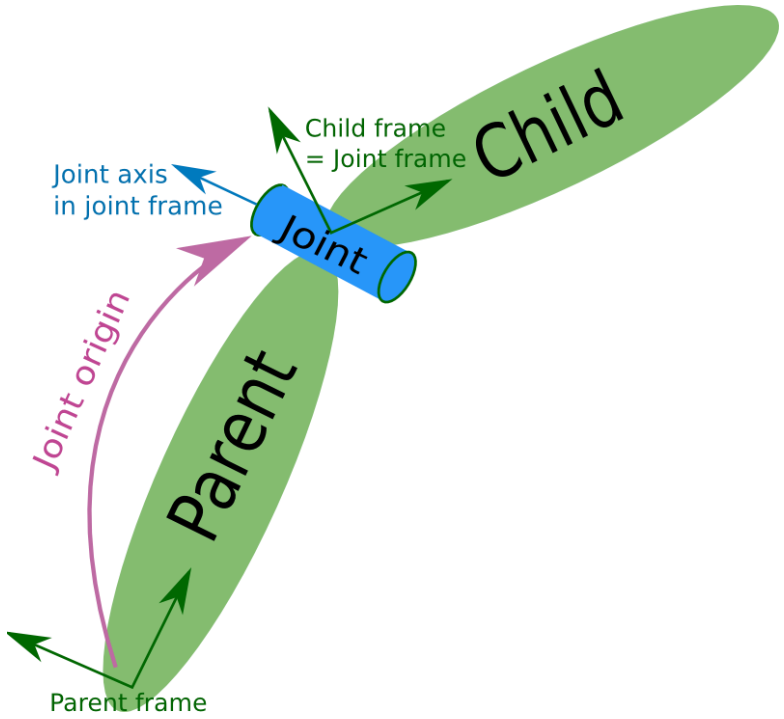

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>

    <dynamics damping="0.0" friction="0.0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```

The joint element describes the kinematics of the joint and also specifies the safety limits of the joint.

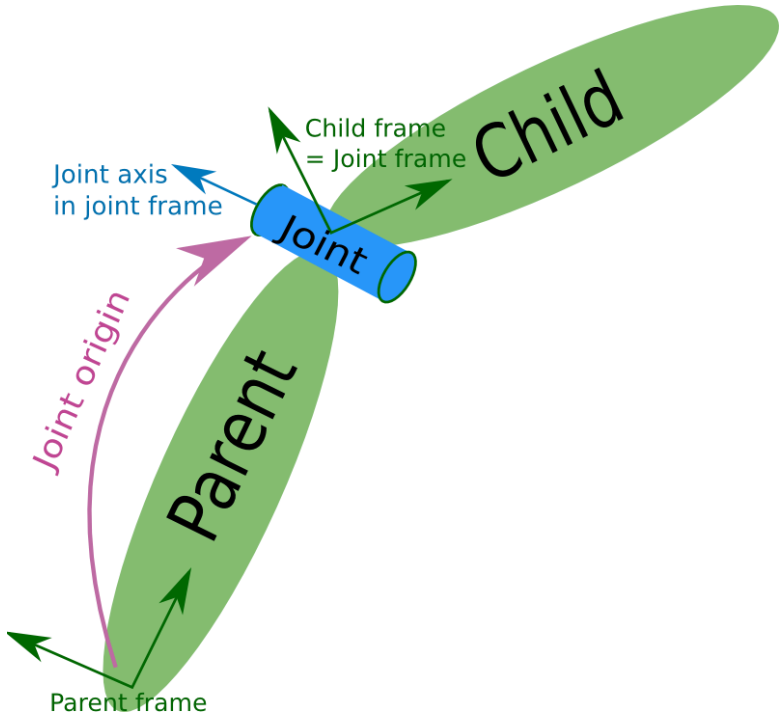
<Joint> element

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <axis xyz="0 1 0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```



name *(required)*
Specifies a unique name of the joint

<Joint> element

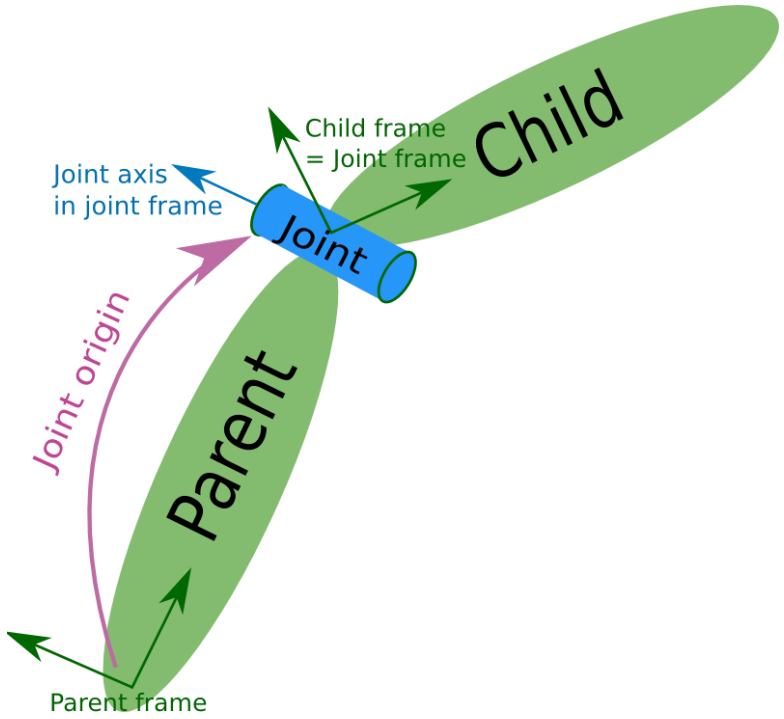



```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <axis xyz="0 1 0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```

type *(required)* → Specifies the **type of joint**, can be one of the following:

- **Revolute** – Hinge joint that rotates along the axis and has a limited range specified by the upper and lower limits.
- **Continuous** – **revolute** with no upper and lower limits.
- **Prismatic** – Sliding joint that slides along the axis, and has a limited range specified by the upper and lower limits.
- **Fixed** – this is not really a joint because it cannot move. All degrees of freedom are locked.
  This type of joint does not require the **<axis>**, **<calibration>**, **<dynamics>**, **<limits>** or **<safety_controller>**.
- **Floating** – Allows motion for all 6 degrees of freedom.
- **Planar** – Allows motion in a plane perpendicular to the axis.

<Joint> element



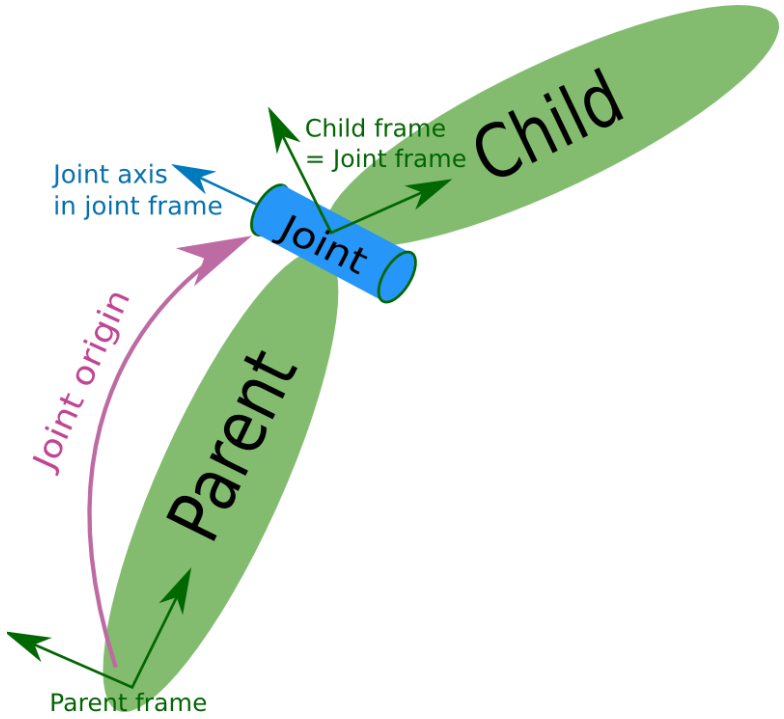

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <axis xyz="0 1 0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```

<origin> *(Defaults to identity if not specified)*
This is the transform from the parent link to the child link.
The joint is located at the origin of the child link, as shown in the figure above.

xyz *(Defaults to zero vector if not specified)*
Represents the ***x, y, z* offset**. All positions are specified in ***metres***.
rpy *(Defaults to zero vector if not specified)*
Represents the rotation around fixed axis: ***roll*, *pitch*, *yaw***. All angles are specified in ***radians***.

<Joint> element

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <axis xyz="0 1 0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```



<parent> *(required)*

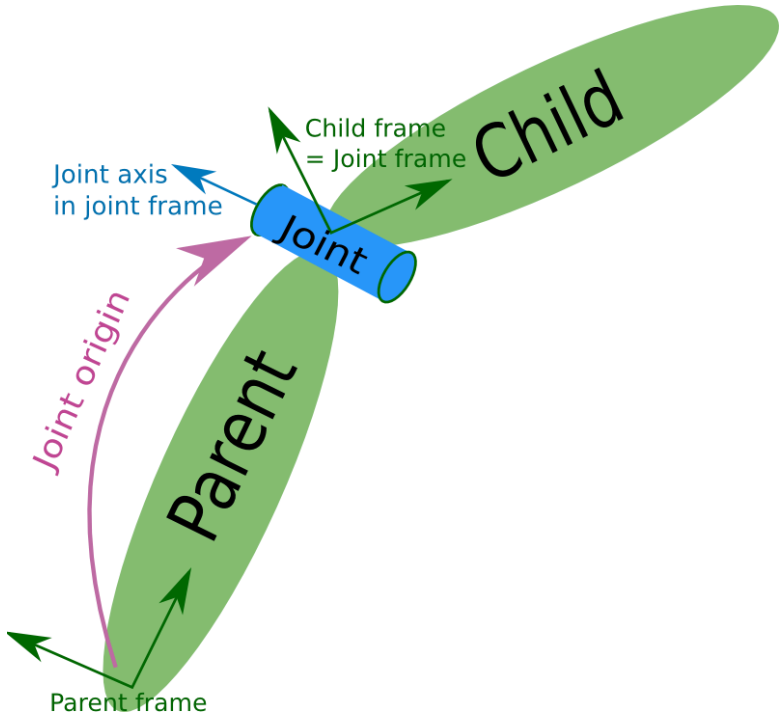
- Parent link name with mandatory attribute:**link**
  - The name of the link that is the parent of this link in the robot tree structure.

<child> *(required)*

- Child link name with mandatory attribute:**link**
  - The name of the link that is the child link.

<Joint> element

```
<joint name="my_joint" type="floating">
    <origin xyz="0 0 1" rpy="0 0 3.1416"/>
    <parent link="link1"/>
    <child link="link2"/>
    <axis xyz="0 1 0"/>
    <limit effort="30" velocity="1.0" lower="-2.2" upper="0.7" />
</joint>
```



<axis> *(optional: defaults to (1,0,0))*
The joint axis specified in the joint frame.

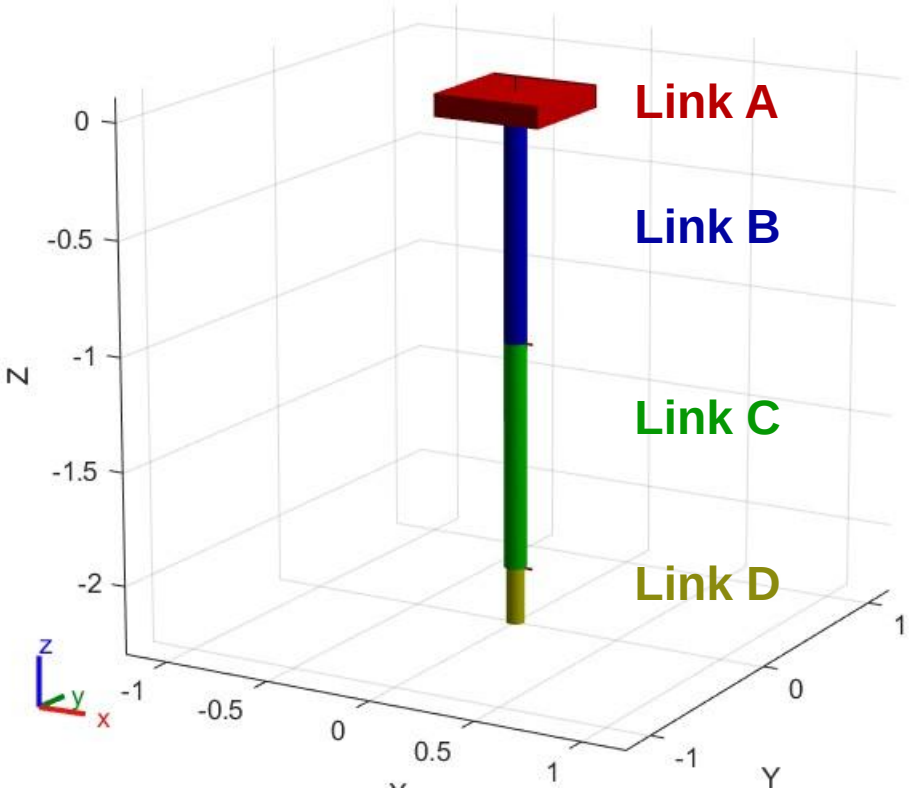
- Axis of rotation for revolute joints,
- Axis of translation for prismatic joints
- Surface normal for planar joints.

The axis is specified in the joint frame of reference. Fixed and floating joints do not use the axis field

The Linkage System

System Overview:

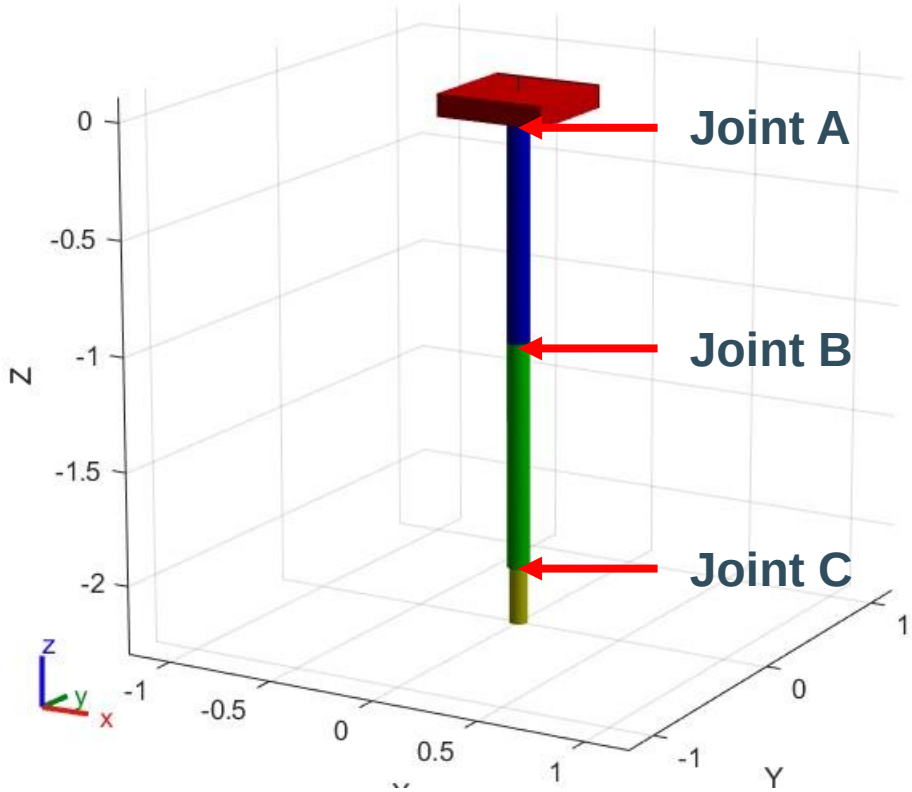
- **Link A**: Base link with a red box geometry
- **Link B**: Blue cylindrical link connected via a revolute joint (Joint A)
- **Link C**: Green cylindrical link connected via another revolute joint (Joint B)
- **Link D**: Yellow link connected via a prismatic joint (Joint C) for linear motion.



The Linkage System

Joint Details:

- **Joint A**: Connects Link A to Link B; rotates around the y-axis.
- **Joint B**: Connects Link B to Link C; rotates around the x-axis.
- **Joint C**: Connects Link C to Link D; allows translation along the z-axis.



The Linkage System

Link A Parameters

- **Mass**: 0.5 kg
- **Inertial Properties**: **Ixx**: 0.005 $kg \cdot m^2$ **Iyy**: 0.005 $kg \cdot m^2$ **Izz**: 0.005 $kg \cdot m^2$
- **Dimensions**: 0.5 x 0.5 x 0.1 m
- **Color**: Red

```
<!-- Link A -->
 <link name="link_A">
   <inertial>
     <origin xyz="0 0 0"/>
     <mass value="0.5"/>
     <inertia ixx="0.005" iyy="0.005" izz="0.005" ixy="0" ixz="0" iyz="0"/>
   </inertial>
</link>
```

The Linkage System

Link A Parameters

- **Mass**: 0.5 kg
- **Inertial Properties**: **Ixx**: 0.005 $kg \cdot m^2$ **Iyy**: 0.005 $kg \cdot m^2$ **Izz**: 0.005 $kg \cdot m^2$
- **Dimensions**: 0.5 x 0.5 x 0.1 m
- **Color**: Red

```
<!-- Link A -->
 <link name="link_A">
<visual>
    <origin xyz="0 0 0"/>
    <geometry>
            <box size="0.5 0.5 0.1"/>
    </geometry>
    <material name="red">
            <color rgba="1 0 0 1"/>
    </material>
</visual>
</link>
```

The Linkage System

Joint A Parameters

- **Type**: Revolute
- **Parent Link**: link_A
- **Child Link**: link_B
- **Origin**:
  - **Position (XYZ)**: 0, 0, -0.05 m
  - **Orientation (RPY)**: 0, 0, 0 rad
- **Axis of Rotation**:
  - **Direction (XYZ)**: 0, 1, 0 (Y-axis)
- **Joint Limits**:
  - **Lower Limit**: -1.57 rad
  - **Upper Limit**: 1.57 rad
  - **Effort**: 10 Nm
  - **Velocity**: 1.0 rad/s

```
<joint name="joint_A" type="revolute">
  <parent link="link_A"/>
  <child link="link_B"/>
  <origin xyz="0 0 -0.05" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```

The Linkage System

Joint A Parameters

- **Type**: Revolute
- **Parent Link**: link_A
- **Child Link**: link_B
- **Origin**:
  - **Position (XYZ)**: 0, 0, -0.05 m
  - **Orientation (RPY)**: 0, 0, 0 rad
- **Axis of Rotation**:
  - **Direction (XYZ)**: 0, 1, 0 (Y-axis)
- **Joint Limits**:
  - **Lower Limit**: -1.57 rad
  - **Upper Limit**: 1.57 rad
  - **Effort**: 10 Nm
  - **Velocity**: 1.0 rad/s

```
<joint name="joint_A" type="revolute">
  <parent link="link_A"/>
  <child link="link_B"/>
  <origin xyz="0 0 -0.05" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```

The Linkage System

Joint A Parameters

- **Type**: Revolute
- **Parent Link**: link_A
- **Child Link**: link_B
- **Origin**:
  - **Position (XYZ)**: 0, 0, -0.05 m
  - **Orientation (RPY)**: 0, 0, 0 rad
- **Axis of Rotation**:
  - **Direction (XYZ)**: 0, 1, 0 (Y-axis)
- **Joint Limits**:
  - **Lower Limit**: -1.57 rad
  - **Upper Limit**: 1.57 rad
  - **Effort**: 10 Nm
  - **Velocity**: 1.0 rad/s

```
<joint name="joint_A" type="revolute">
  <parent link="link_A"/>
  <child link="link_B"/>
  <origin xyz="0 0 -0.05" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```

The Linkage System

Joint A Parameters

- **Type**: Revolute
- **Parent Link**: link_A
- **Child Link**: link_B
- **Origin**:
  - **Position (XYZ)**: 0, 0, -0.05 m
  - **Orientation (RPY)**: 0, 0, 0 rad
- **Axis of Rotation**:
  - **Direction (XYZ)**: 0, 1, 0 (Y-axis)
- **Joint Limits**:
  - **Lower Limit**: -1.57 rad
  - **Upper Limit**: 1.57 rad
  - **Effort**: 10 Nm
  - **Velocity**: 1.0 rad/s

```
<joint name="joint_A" type="revolute">
  <parent link="link_A"/>
  <child link="link_B"/>
  <origin xyz="0 0 -0.05" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```

The Linkage System

Joint A Parameters

- **Type**: Revolute
- **Parent Link**: link_A
- **Child Link**: link_B
- **Origin**:
  - **Position (XYZ)**: 0, 0, -0.05 m
  - **Orientation (RPY)**: 0, 0, 0 rad
- **Axis of Rotation**:
  - **Direction (XYZ)**: 0, 1, 0 (Y-axis)
- **Joint Limits**:
  - **Lower Limit**: -1.57 rad
  - **Upper Limit**: 1.57 rad
  - **Effort**: 10 Nm
  - **Velocity**: 1.0 rad/s

```
<joint name="joint_A" type="revolute">
  <parent link="link_A"/>
  <child link="link_B"/>
  <origin xyz="0 0 -0.05" rpy="0 0 0"/>
  <axis xyz="0 1 0"/>
  <limit lower="-1.57" upper="1.57" effort="10" velocity="1.0"/>
</joint>
```

Exercise: Design a 2-DOF Robot with URDF

Objective:
The robot operates in the 2D X-Y plane, so you will define the robot's joints and links accordingly

Joints Configuration:
The robot should have two joints. These joints can be:

- Both *revolute*
- Both *prismatic*
- A combination of one *revolute* and one *prismatic*

You are free to choose the configuration that you prefer but document the type of joints you have selected

Link Specifications:

- **Link 1 (L1):** Length = **0.6 m**
- **Link 2 (L2):** Length = **0.4 m**

You may represent the links as simple shapes (e.g., boxes or cylinders).
These shapes will be used to calculate the robot's physical properties.

Exercise: Design a 2-DOF Robot with URDF

URDF File:
Define the **URDF file** with the following elements:

- **Links:** Define two links with the specified lengths.
- **Joints:** Define the two joints, including their types (revolute, prismatic, or mixed).
- **Physical properties:** Include the mass, inertia, and center of mass for each link
- **Visual and Collision Properties:** Provide visual and collision shapes for the links (using simple geometric representations like boxes or cylinders).

Exercise: Design a 2-DOF Robot with URDF

Deliverables:

1. URDF File:

A well-formed URDF file representing the 2-DOF robot. Ensure that the file includes the correct joint types, link dimensions, and physical properties.

2. Documentation:

A brief report describing the design of your robot, including:

- The choice of joint types (revolute, prismatic, or mixed) and why you selected them.
- The mass, inertia, and center of mass calculations for each link.

Exercise: Design a 2-DOF Robot with URDF

Hints:

- **Inertia Calculation:**

For simple shapes like rods or boxes, you can find the mass moment of inertia from standard formulas (e.g., for a uniform rod, $I = \frac{1}{12} ML^2$ about its center of mass).

- **Reference URDF File:**

You will be provided with a reference URDF file. Please use this file as a template and/or modify it as needed to meet the specified requirements.

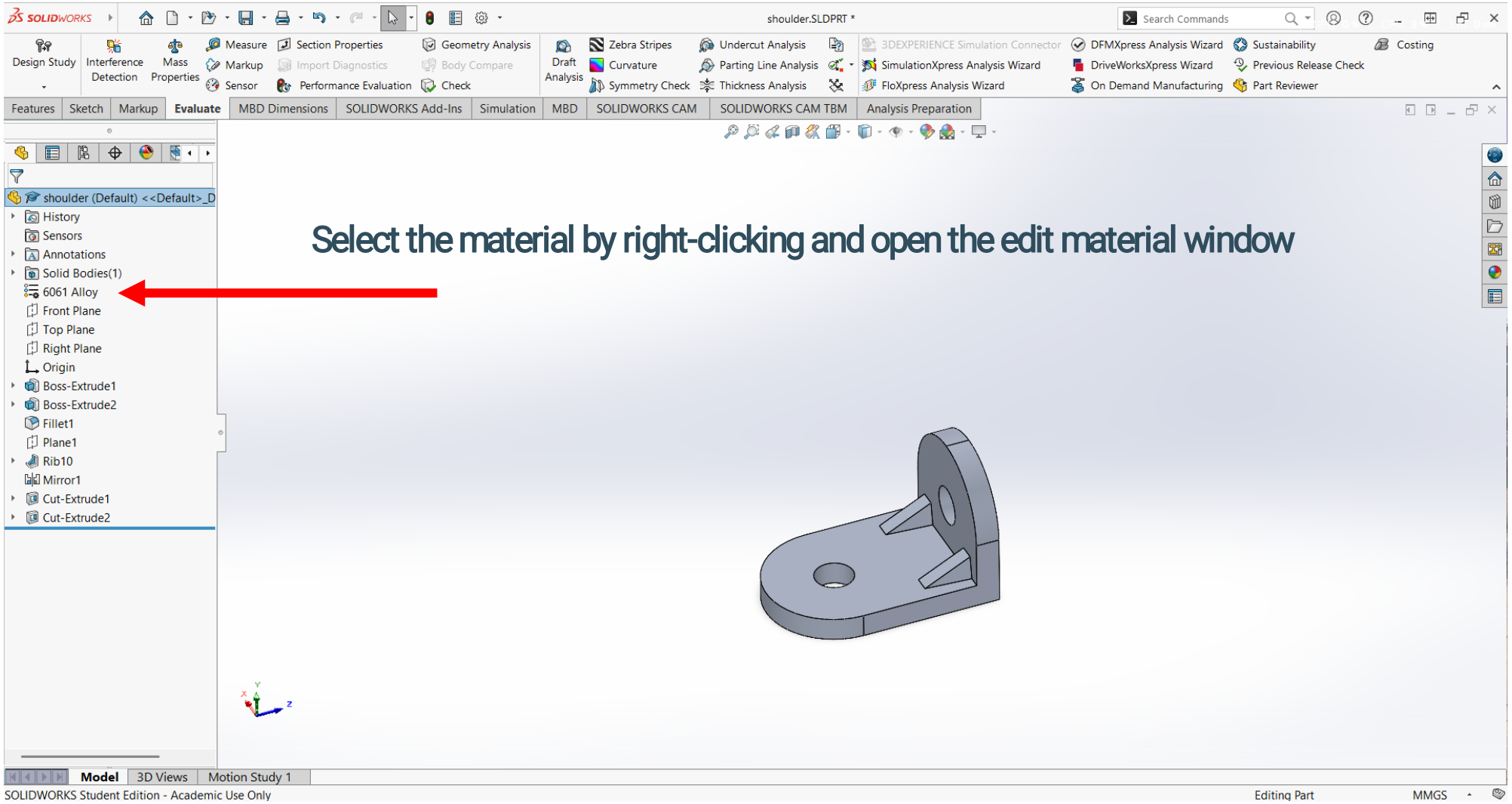
Bonus

While simple shapes can be easily represented in a URDF file, real-world robots often require more complex geometry. Here's how you can include complex parts in your URDF:

Steps to Include Complex Geometry:

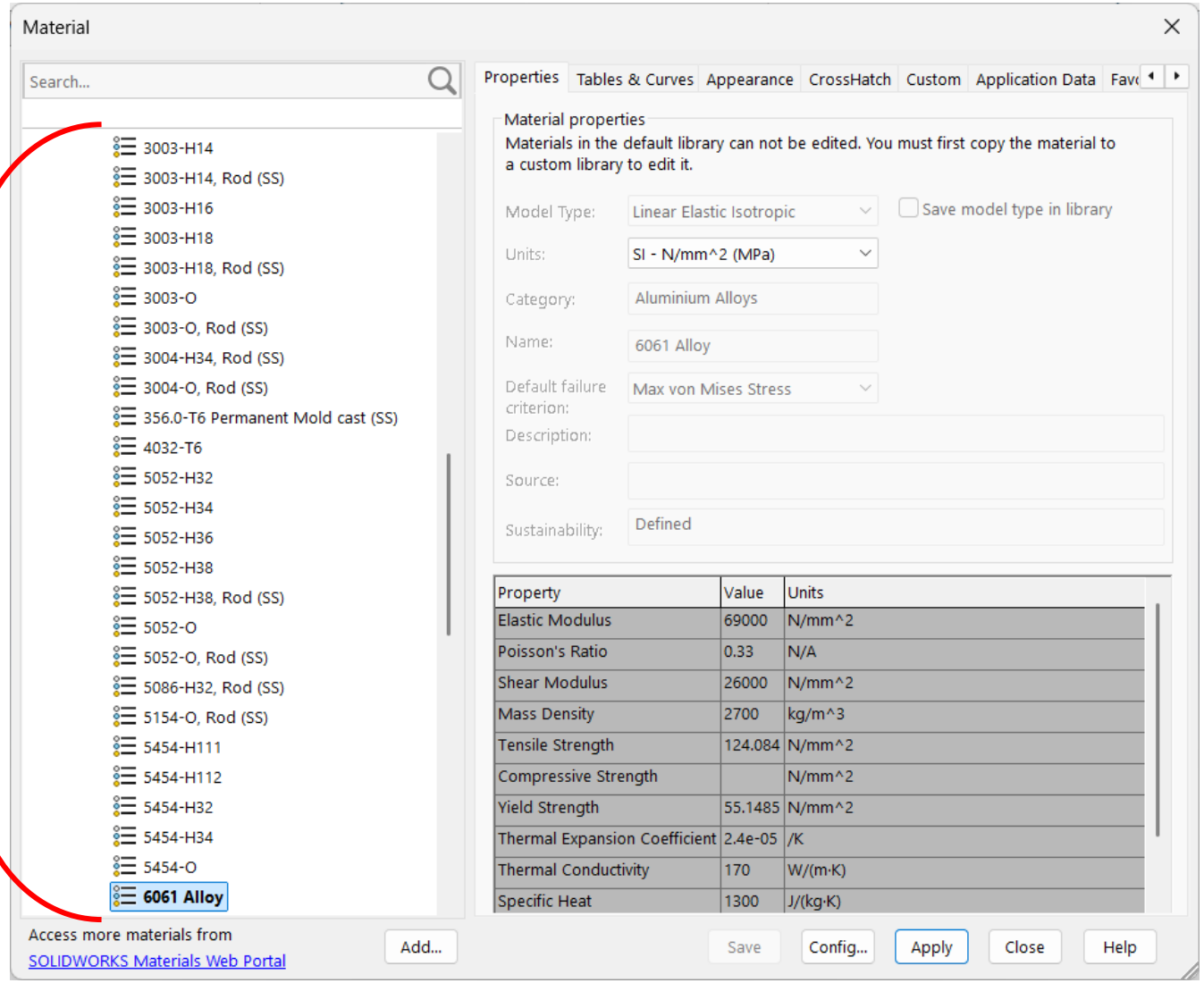
1. Design the Part Using 3D Software:
    - Use software like **SolidWorks** to create detailed and complex 3D models of robotic components
2. **Export the Model:**
    - **.STL** (for mesh geometry)
    - **.DAE** (for more advanced models with textures or materials)
3. Include Inertial Properties:
    - You can calculate the **mass** and **inertia** properties of your part in SolidWorks.
4. Import into URDF

Bonus

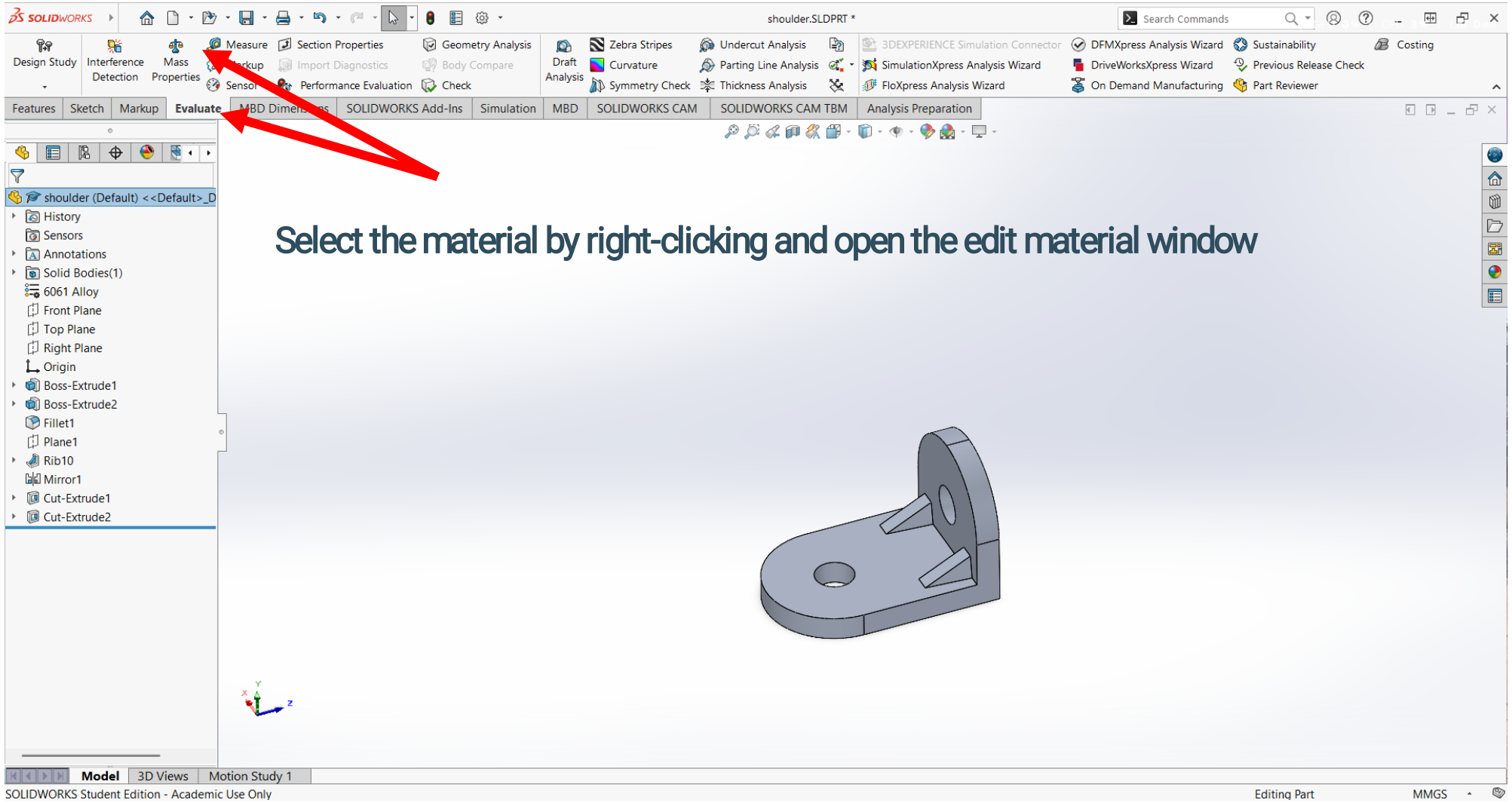

shoulder.SLDPRT *
Search Commands
Design Study
Interference Detection
Mass Properties
Measure
Markup
Sensor
Section Properties
Import Diagnostics
Performance Evaluation
Geometry Analysis
Body Compare
Check
Draft Analysis
Zebra Stripes
Curvature
Symmetry Check
Undercut Analysis
Parting Line Analysis
Thickness Analysis
3DEXPERIENCE Simulation Connector
SimulationXpress Analysis Wizard
FloXpress Analysis Wizard
DFMXpress Analysis Wizard
DriveWorksXpress Wizard
On Demand Manufacturing
Sustainability
Previous Release Check
Part Reviewer
Costing
Features
Sketch
Markup
Evaluate
MBD Dimensions
SOLIDWORKS Add-Ins
Simulation
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation
shoulder (Default) <<Default>_D
History
Sensors
Annotations
Solid Bodies(1)
6061 Alloy
Front Plane
Top Plane
Right Plane
Origin
Boss-Extrude1
Boss-Extrude2
Fillet1
Plane1
Rib10
Mirror1
Cut-Extrude1
Cut-Extrude2
Select the material by right-clicking and open the edit material window
Model
3D Views
Motion Study 1
SOLIDWORKS Student Edition - Academic Use Only
Editing Part
MMGS

Bonus

You can select different a vast list of materials like steel, wood, titanium etc



Bonus

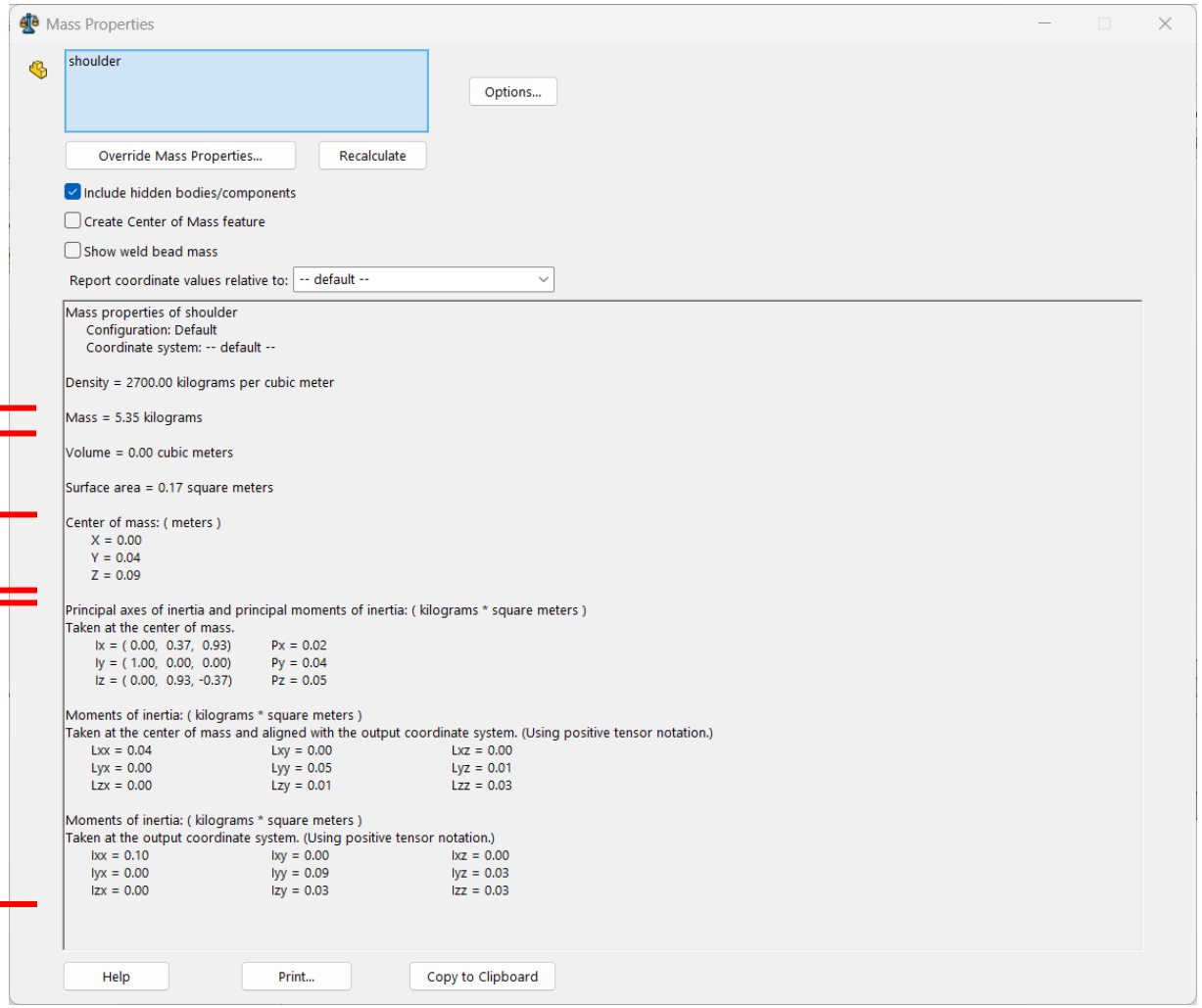

shoulder.SLDPRT *
Search Commands
Design Study
Interference Detection
Mass Properties
Measure
Section Properties
Geometry Analysis
Import Diagnostics
Body Compare
Performance Evaluation
Check
Draft Analysis
Zebra Stripes
Curvature
Symmetry Check
Undercut Analysis
Parting Line Analysis
Thickness Analysis
3DEXPERIENCE Simulation Connector
SimulationXpress Analysis Wizard
FloXpress Analysis Wizard
DFMXpress Analysis Wizard
DriveWorksXpress Wizard
On Demand Manufacturing
Sustainability
Previous Release Check
Part Reviewer
Costing
Features
Sketch
Markup
Evaluate
SOLIDWORKS Add-Ins
Simulation
MBD
SOLIDWORKS CAM
SOLIDWORKS CAM TBM
Analysis Preparation
shoulder (Default) <<Default>_D
History
Sensors
Annotations
Solid Bodies(1)
6061 Alloy
Front Plane
Top Plane
Right Plane
Origin
Boss-Extrude1
Boss-Extrude2
Fillet1
Plane1
Rib10
Mirror1
Cut-Extrude1
Cut-Extrude2
Select the material by right-clicking and open the edit material window
Model
3D Views
Motion Study 1
SOLIDWORKS Student Edition - Academic Use Only
Editing Part
MMGS

Bonus

Mass

Location of the center of mass

inertia



Bonus

Make sure to specify the correct file path when importing the STL file into the URDF

```
<visual>
    <origin rpy="0 0 0" xyz="0 0 0"/>
    <geometry>
      <mesh filename="package://robot_description/meshes/base_link.STL"/>
    </geometry>
  </visual>
```