urdfpy Documentation

Matthew Matl

Apr 24, 2019

Contents

Urdfpy is a simple pure-Python library for loading, manipulating, and exporting URDF files and robot specifications. For example, here's a rendering of a UR5 robot moving around after being loaded by this library.

CHAPTER 1

Installation and Setup Guide

1.1 Python Installation

This package is pip-installable for any Python version. Simply run the following command:

```
pip install urdfpy
```

1.2 Installing in Development Mode

If you're planning on contributing to this repository, please see the *Development Guide*.

CHAPTER 2

Usage Examples

This page documents several simple use cases for you to try out. For full details, see the *API Reference*, and check out the full class reference for *URDF*.

2.1 Loading from a File

You can load a URDF from any `.urdf` file, as long as you fix the links to be relative or absolute links rather than ROS resource URLs.

```
>>> from urdfpy import URDF
>>> robot = URDF.load('tests/data/ur5/ur5.urdf')
```

2.2 Saving to a File

You can also export the URDF to a file. Any meshes and images will be dumped using their original relative or absolute names. If the names are relative, they'll be dumped relative to the new URDF file.

```
>>> robot.save('/tmp/ur5/ur5.urdf')
```

2.3 Accessing Links and Joints

You have direct access to link and joint information.

```
>>> for link in robot.links:
...     print(link.name)
...
base_link
shoulder_link
```

(continues on next page)

(continued from previous page)

```
upper_arm_link
forearm_link
wrist_1_link
wrist_2_link
wrist_3_link
ee_link
base
tool0
world
```

```
>>> for joint in robot.joints:
...     print('{} connects {} to {}'.format(
...         joint.name, joint.parent, joint.child
...     ))
...
shoulder_pan_joint connects base_link to shoulder_link
shoulder_lift_joint connects shoulder_link to upper_arm_link
elbow_joint connects upper_arm_link to forearm_link
wrist_1_joint connects forearm_link to wrist_1_link
wrist_2_joint connects wrist_1_link to wrist_2_link
wrist_3_joint connects wrist_2_link to wrist_3_link
ee_fixed_joint connects wrist_3_link to ee_link
base_link-base_fixed_joint connects base_link to base
wrist_3_link-tool0_fixed_joint connects wrist_3_link to tool0
world_joint connects world to base_link
```

You can also access which joints can be articulated:

```
>>> for joint in robot.actuated_joints:
...     print(joint.name)
...
shoulder_pan_joint
shoulder_lift_joint
elbow_joint
wrist_1_joint
wrist_2_joint
wrist_3_joint
```

And also which link is the base link:

```
>>> print(robot.base_link.name)
world
```

2.4 Doing Forward Kinematics

You have a variety of options for performing forward kinematics. For example, you can get the kinematics of the robot's links:

```
>>> fk = robot.link_fk()
>>> print(fk[robot.links[0]])
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
```

(continues on next page)

(continued from previous page)

```
       [0., 0., 0., 1.]])
>>> print(fk[robot.links[1]])
array([[1.   , 0.   , 0.   , 0.   ],
       [0.   , 1.   , 0.   , 0.   ],
       [0.   , 0.   , 1.   , 0.089],
       [0.   , 0.   , 0.   , 1.   ]])
>>> fk = robot.link_fk(cfg={'shoulder_pan_joint' : 1.0})
>>> print(fk[robot.links[1]])
array([[ 0.54 , -0.841,  0.   ,  0.   ],
       [ 0.841,  0.54 ,  0.   ,  0.   ],
       [ 0.   ,  0.   ,  1.   ,  0.089],
       [ 0.   ,  0.   ,  0.   ,  1.   ]])
```

The `fk` result is a map from `Link` objects to their poses relative to the robot's base link as 4x4 homogenous transform matrices. You can pass a joint configuration, which is a map from joints (or joint names) to joint configuration values.

You can also directly get the poses of the robot's `Trimesh` geometries:

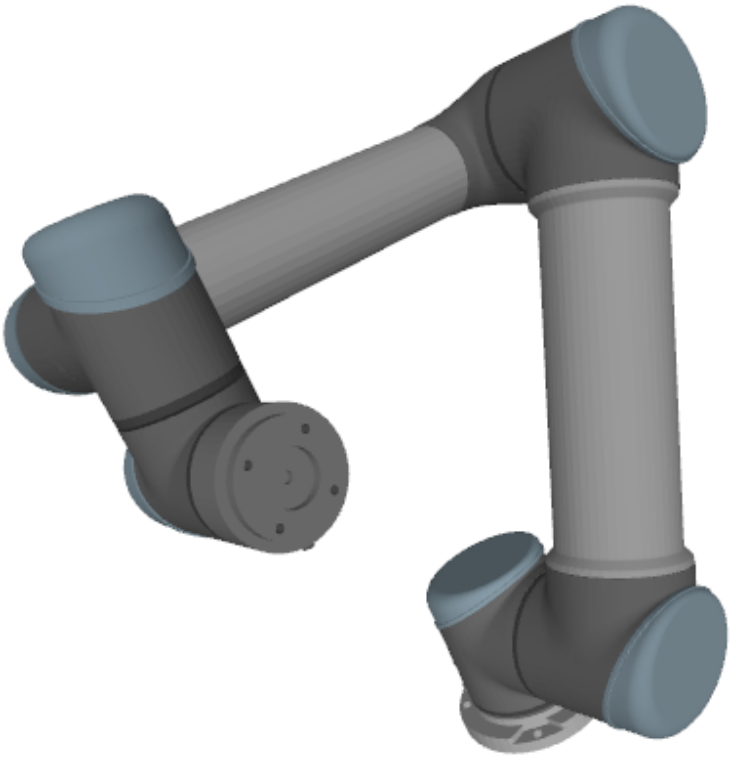
```
>>> fk = robot.visual_trimesh_fk()
>>> print(type(list(fk.keys())[0]))
trimesh.base.Trimesh
>>> fk = robot.collision_trimesh_fk()
>>> print(type(list(fk.keys())[0]))
trimesh.base.Trimesh
```

2.5 Visualization

Urdfpy also comes bundled with two simple visualization functions.

You can visualize a robot in a static configuration:

```
>>> robot.show(cfg={
...     'shoulder_lift_joint': -2.0,
...     'elbow_joint': 2.0
... })
...
```



Or animate it over a configuration trajectory:

```
>>> robot.animate(cfg_trajectory={
...     'shoulder_pan_joint' : [-np.pi / 4, np.pi / 4],
...     'shoulder_lift_joint' : [0.0, -np.pi / 2.0],
...     'elbow_joint' : [0.0, np.pi / 2.0]
... })
...
```

CHAPTER 3

API Reference

3.1 Functions

rpy_to_matrix(coords)	Convert roll-pitch-yaw coordinates to a 3x3 homogenous rotation matrix.
matrix_to_rpy(R[, solution])	Convert a 3x3 transform matrix to roll-pitch-yaw coordinates.
xyz_rpy_to_matrix(xyz_rpy)	Convert xyz_rpy coordinates to a 4x4 homogenous matrix.
matrix_to_xyz_rpy(matrix)	Convert a 4x4 homogenous matrix to xyzrpy coordinates.

3.1.1 rpy_to_matrix

urdfpy.**rpy_to_matrix**(*coords*)

Convert roll-pitch-yaw coordinates to a 3x3 homogenous rotation matrix.

The roll-pitch-yaw axes in a typical URDF are defined as a rotation of `r` radians around the x-axis followed by a rotation of `p` radians around the y-axis followed by a rotation of `y` radians around the z-axis. These are the Z1-Y2-X3 Tait-Bryan angles. See Wikipedia for more information.

Parameters **coords** (*(3,) float*) – The roll-pitch-yaw coordinates in order (x-rot, y-rot, z-rot).

Returns **R** – The corresponding homogenous 3x3 rotation matrix.

Return type (3,3) float

3.1.2 matrix_to_rpy

urdfpy.**matrix_to_rpy**(*R*, *solution=1*)

Convert a 3x3 transform matrix to roll-pitch-yaw coordinates.

The roll-pitchRyaw axes in a typical URDF are defined as a rotation of `r` radians around the x-axis followed by a rotation of `p` radians around the y-axis followed by a rotation of `y` radians around the z-axis. These are the Z1-Y2-X3 Tait-Bryan angles. See Wikipedia for more information.

There are typically two possible roll-pitch-yaw coordinates that could have created a given rotation matrix. Specify `solution=1` for the first one and `solution=2` for the second one.

Parameters

- **R** (*(3, 3) float*) – A 3x3 homogenous rotation matrix.
- **solution** (*int*) – Either 1 or 2, indicating which solution to return.

Returns **coords** – The roll-pitch-yaw coordinates in order (x-rot, y-rot, z-rot).

Return type (3,) float

3.1.3 xyz_rpy_to_matrix

urdfpy.**xyz_rpy_to_matrix**(*xyz_rpy*)

Convert xyz_rpy coordinates to a 4x4 homogenous matrix.

Parameters **xyz_rpy** (*(6,) float*) – The xyz_rpy vector.

Returns **matrix** – The homogenous transform matrix.

Return type (4,4) float

3.1.4 matrix_to_xyz_rpy

urdfpy.**matrix_to_xyz_rpy**(*matrix*)

Convert a 4x4 homogenous matrix to xyzrpy coordinates.

Parameters **matrix** (*(4, 4) float*) – The homogenous transform matrix.

Returns **xyz_rpy** – The xyz_rpy vector.

Return type (6,) float

3.2 Classes

URDFType()	Abstract base class for all URDF types.
Box(size)	A rectangular prism whose center is at the local origin.
Cylinder(radius, length)	A cylinder whose center is at the local origin.
Sphere(radius)	A sphere whose center is at the local origin.
Mesh(filename[, scale, meshes])	A triangular mesh object.
Geometry([box, cylinder, sphere, mesh])	A wrapper for all geometry types.
Texture(filename[, image])	An image-based texture.
Material(name[, color, texture])	A material for some geometry.
Collision(name, origin, geometry)	Collision properties of a link.
Visual(geometry[, name, origin, material])	Visual properties of a link.
Inertial(mass, inertia[, origin])	The inertial properties of a link.
JointCalibration([rising, falling])	The reference positions of the joint.
JointDynamics(damping, friction)	The dynamic properties of the joint.

Continued on next page

Table 2 – continued from previous page

JointLimit(effort, velocity[, lower, upper])	The limits of the joint.
JointMimic(joint[, multiplier, offset])	A mimicry tag for a joint, which forces its configuration to mimic another joint's.
SafetyController(k_velocity[, k_position, ...])	A controller for joint movement safety.
Actuator(name[, mechanicalReduction, ...])	An actuator.
TransmissionJoint(name, hardwareInterfaces)	A transmission joint specification.
Transmission(name, trans_type[, joints, ...])	An element that describes the relationship between an actuator and a joint.
Joint(name, joint_type, parent, child[, ...])	A connection between two links.
Link(name, inertial, visuals, collisions)	A link of a rigid object.
URDF(name, links[, joints, transmissions, ...])	The top-level URDF specification.

3.2.1 URDFType

class urdfpy.**URDFType**

Bases: object

Abstract base class for all URDF types.

This has useful class methods for automatic parsing/unparsing of XML trees.

There are three overridable class variables:

- _ATTRIBS - This is a dictionary mapping attribute names to a tuple, `(type, required)` where `type` is the Python type for the attribute and `required` is a boolean stating whether the attribute is required to be present.
- _ELEMENTS - This is a dictionary mapping element names to a tuple, `(type, required, multiple)` where `type` is the Python type for the element, `required` is a boolean stating whether the element is required to be present, and `multiple` is a boolean indicating whether multiple elements of this type could be present. Elements are child nodes in the XML tree, and their type must be a subclass of *URDFType*.
- _TAG - This is a string that represents the XML tag for the node containing this type of object.

3.2.2 Box

class urdfpy.**Box**(*size*)

Bases: *urdfpy.URDFType*

A rectangular prism whose center is at the local origin.

Parameters **size** (*(3,) float*) – The length, width, and height of the box in meters.

Attributes Summary

meshes	The triangular meshes that represent this object.
size	The length, width, and height of the box in meters.

Attributes Documentation

meshes

The triangular meshes that represent this object.

Type list of Trimesh

size
The length, width, and height of the box in meters.

Type (3,) float

3.2.3 Cylinder

class urdfpy.**Cylinder**(*radius*, *length*)
Bases: *urdfpy.URDFType*

A cylinder whose center is at the local origin.

Parameters

- **radius** (*float*) – The radius of the cylinder in meters.
- **length** (*float*) – The length of the cylinder in meters.

Attributes Summary

length	The length of the cylinder in meters.
meshes	The triangular meshes that represent this object.
radius	The radius of the cylinder in meters.

Attributes Documentation

length
The length of the cylinder in meters.

Type float

meshes
The triangular meshes that represent this object.

Type list of Trimesh

radius
The radius of the cylinder in meters.

Type float

3.2.4 Sphere

class urdfpy.**Sphere**(*radius*)
Bases: *urdfpy.URDFType*

A sphere whose center is at the local origin.

Parameters **radius** (*float*) – The radius of the sphere in meters.

Attributes Summary

meshes	The triangular meshes that represent this object.
radius	The radius of the sphere in meters.

Attributes Documentation

meshes
The triangular meshes that represent this object.

Type list of Trimesh

radius
The radius of the sphere in meters.

Type float

3.2.5 Mesh

class urdfpy.**Mesh**(*filename*, *scale=None*, *meshes=None*)
Bases: *urdfpy.URDFType*

A triangular mesh object.

Parameters

- **filename** (*str*) – The path to the mesh that contains this object. This can be relative to the top-level URDF or an absolute path.
- **scale** (*(3,) float, optional*) – The scaling value for the mesh along the XYZ axes. If None, assumes no scale is applied.
- **meshes** (list of Trimesh) – A list of meshes that compose this mesh. The list of meshes is useful for visual geometries that might be composed of separate trimesh objects. If not specified, the mesh is loaded from the file using trimesh.

Attributes Summary

filename	The path to the mesh file for this object.
meshes	The triangular meshes that represent this object.
scale	A scaling for the mesh along its local XYZ axes.

Attributes Documentation

filename
The path to the mesh file for this object.

Type str

meshes
The triangular meshes that represent this object.

Type list of Trimesh

scale
A scaling for the mesh along its local XYZ axes.

Type (3,) float

3.2.6 Geometry

class urdfpy.**Geometry**(*box=None*, *cylinder=None*, *sphere=None*, *mesh=None*)
Bases: *urdfpy.URDFType*

A wrapper for all geometry types.

Only one of the following values can be set, all others should be set to `None`.

Parameters

- **box** (*Box*, optional) – Box geometry.
- **cylinder** (*Cylinder*) – Cylindrical geometry.
- **sphere** (*Sphere*) – Spherical geometry.
- **mesh** (*Mesh*) – Mesh geometry.

Attributes Summary

box	Box geometry.
cylinder	Cylinder geometry.
geometry	*Box*, *Cylinder*, *Sphere*, or *Mesh* : The valid geometry element.
mesh	Mesh geometry.
meshes	The geometry's triangular mesh representation(s).
sphere	Spherical geometry.

Attributes Documentation

box
Box geometry.

Type *Box*

cylinder
Cylinder geometry.

Type *Cylinder*

geometry
Box, *Cylinder*, *Sphere*, or *Mesh* : The valid geometry element.

mesh
Mesh geometry.

Type *Mesh*

meshes
The geometry's triangular mesh representation(s).

Type list of `Trimesh`

sphere
Spherical geometry.

Type *Sphere*

3.2.7 Texture

class urdfpy.**Texture**(*filename*, *image=None*)
Bases: *urdfpy.URDFType*

An image-based texture.

Parameters

- **filename** (*str*) – The path to the image that contains this texture. This can be relative to the top-level URDF or an absolute path.
- **image** (PIL.Image.Image, optional) – The image for the texture. If not specified, it is loaded automatically from the filename.

Attributes Summary

filename	Path to the image for this texture.
image	The image for this texture.

Attributes Documentation

filename
Path to the image for this texture.

Type str

image
The image for this texture.

Type PIL.Image.Image

3.2.8 Material

class urdfpy.**Material**(*name*, *color=None*, *texture=None*)
Bases: *urdfpy.URDFType*

A material for some geometry.

Parameters

- **name** (*str*) – The name of the material.
- **color** (*(4,) float, optional*) – The RGBA color of the material in the range [0,1].
- **texture** (*Texture*, optional) – A texture for the material.

Attributes Summary

color	The RGBA color of the material, in the range [0,1].
name	The name of the material.
texture	The texture for the material.

Attributes Documentation

color
The RGBA color of the material, in the range [0,1].

Type (4,) float

name
The name of the material.

Type str

texture
The texture for the material.

Type *Texture*

3.2.9 Collision

class urdfpy.**Collision**(*name*, *origin*, *geometry*)
Bases: *urdfpy.URDFType*

Collision properties of a link.

Parameters

- **geometry** (*Geometry*) – The geometry of the element
- **name** (*str, optional*) – The name of the collision geometry.
- **origin** (*(4,4) float, optional*) – The pose of the collision element relative to the link frame. Defaults to identity.

Attributes Summary

geometry	The geometry of this element.
name	The name of this collision element.
origin	The pose of this element relative to the link frame.

Attributes Documentation

geometry
The geometry of this element.

Type *Geometry*

name
The name of this collision element.

Type str

origin
The pose of this element relative to the link frame.

Type (4,4) float

3.2.10 Visual

class urdfpy.**Visual**(*geometry*, *name=None*, *origin=None*, *material=None*)
Bases: *urdfpy.URDFType*

Visual properties of a link.

Parameters

- **geometry** (*Geometry*) – The geometry of the element
- **name** (*str, optional*) – The name of the visual geometry.
- **origin** (*(4,4) float, optional*) – The pose of the visual element relative to the link frame. Defaults to identity.
- **material** (*Material*, optional) – The material of the element.

Attributes Summary

geometry	The geometry of this element.
material	The material for this element.
name	The name of this visual element.
origin	The pose of this element relative to the link frame.

Attributes Documentation

geometry
The geometry of this element.

Type *Geometry*

material
The material for this element.

Type *Material*

name
The name of this visual element.

Type str

origin
The pose of this element relative to the link frame.

Type (4,4) float

3.2.11 Inertial

class urdfpy.**Inertial**(*mass*, *inertia*, *origin=None*)
Bases: *urdfpy.URDFType*

The inertial properties of a link.

Parameters

- **mass** (*float*) – The mass of the link in kilograms.
- **inertia** (*(3,3) float*) – The 3x3 symmetric rotational inertia matrix.

- **origin** (*(4,4) float, optional*) – The pose of the inertials relative to the link frame. Defaults to identity if not specified.

Attributes Summary

inertia	The 3x3 symmetric rotational inertia matrix.
mass	The mass of the link in kilograms.
origin	The pose of the inertials relative to the link frame.

Attributes Documentation

inertia
The 3x3 symmetric rotational inertia matrix.

Type (3,3) float

mass
The mass of the link in kilograms.

Type float

origin
The pose of the inertials relative to the link frame.

Type (4,4) float

3.2.12 JointCalibration

class urdfpy.**JointCalibration**(*rising=None*, *falling=None*)
Bases: *urdfpy.URDFType*

The reference positions of the joint.

Parameters

- **rising** (*float, optional*) – When the joint moves in a positive direction, this position will trigger a rising edge.
- **falling** – When the joint moves in a positive direction, this position will trigger a falling edge.

Attributes Summary

falling	description.
rising	description.

Attributes Documentation

falling
description.

Type float

rising
description.

Type float

3.2.13 JointDynamics

class urdfpy.**JointDynamics**(*damping*, *friction*)
Bases: *urdfpy.URDFType*

The dynamic properties of the joint.

Parameters

- **damping** (*float*) – The damping value of the joint (Ns/m for prismatic joints, Nms/rad for revolute).
- **friction** (*float*) – The static friction value of the joint (N for prismatic joints, Nm for revolute).

Attributes Summary

damping	The damping value of the joint.
friction	The static friction value of the joint.

Attributes Documentation

damping
The damping value of the joint.

Type float

friction
The static friction value of the joint.

Type float

3.2.14 JointLimit

class urdfpy.**JointLimit**(*effort*, *velocity*, *lower=None*, *upper=None*)
Bases: *urdfpy.URDFType*

The limits of the joint.

Parameters

- **effort** (*float*) – The maximum joint effort (N for prismatic joints, Nm for revolute).
- **velocity** (*float*) – The maximum joint velocity (m/s for prismatic joints, rad/s for revolute).
- **lower** (*float, optional*) – The lower joint limit (m for prismatic joints, rad for revolute).
- **upper** (*float, optional*) – The upper joint limit (m for prismatic joints, rad for revolute).

Attributes Summary

effort	The maximum joint effort.
lower	The lower joint limit.
upper	The upper joint limit.
velocity	The maximum joint velocity.

Attributes Documentation

effort
The maximum joint effort.

Type float

lower
The lower joint limit.

Type float

upper
The upper joint limit.

Type float

velocity
The maximum joint velocity.

Type float

3.2.15 JointMimic

class urdfpy.**JointMimic**(*joint*, *multiplier=None*, *offset=None*)
Bases: *urdfpy.URDFType*

A mimicry tag for a joint, which forces its configuration to mimic another joint's.

This joint's configuration value is set equal to `multiplier * other_joint_cfg + offset`.

Parameters

- **joint** (*str*) – The name of the joint to mimic.
- **multiplier** (*float*) – The joint configuration multiplier. Defaults to 1.0.
- **offset** (*float, optional*) – The joint configuration offset. Defaults to 0.0.

Attributes Summary

joint	The name of the joint to mimic.
multiplier	The multiplier for the joint configuration.
offset	The offset for the joint configuration

Attributes Documentation

joint
The name of the joint to mimic.

Type float

multiplier
The multiplier for the joint configuration.

Type float

offset
The offset for the joint configuration

Type float

3.2.16 SafetyController

class urdfpy.**SafetyController**(*k_velocity*, *k_position=None*, *soft_lower_limit=None*, *soft_upper_limit=None*)

Bases: *urdfpy.URDFType*

A controller for joint movement safety.

Parameters

- **k_velocity** (*float*) – An attribute specifying the relation between the effort and velocity limits.
- **k_position** (*float, optional*) – An attribute specifying the relation between the position and velocity limits. Defaults to 0.0.
- **soft_lower_limit** (*float, optional*) – The lower joint boundary where the safety controller kicks in. Defaults to 0.0.
- **soft_upper_limit** (*float, optional*) – The upper joint boundary where the safety controller kicks in. Defaults to 0.0.

Attributes Summary

k_position	A relation between the position and velocity limits.
k_velocity	A relation between the effort and velocity limits.
soft_lower_limit	The soft lower limit where the safety controller kicks in.
soft_upper_limit	The soft upper limit where the safety controller kicks in.

Attributes Documentation

k_position
A relation between the position and velocity limits.

Type float

k_velocity
A relation between the effort and velocity limits.

Type float

soft_lower_limit
The soft lower limit where the safety controller kicks in.

Type float

soft_upper_limit
The soft upper limit where the safety controller kicks in.

Type float

3.2.17 Actuator

class urdfpy.**Actuator**(*name*, *mechanicalReduction=None*, *hardwareInterfaces=None*)
Bases: *urdfpy.URDFType*

An actuator.

Parameters

- **name** (*str*) – The name of this actuator.
- **mechanicalReduction** (*str, optional*) – A specifier for the mechanical reduction at the joint/actuator transmission.
- **hardwareInterfaces** (*list of str, optional*) – The supported hardware interfaces to the actuator.

Attributes Summary

hardwareInterfaces	The supported hardware interfaces.
mechanicalReduction	A specifier for the type of mechanical reduction.
name	The name of this actuator.

Attributes Documentation

hardwareInterfaces
The supported hardware interfaces.

Type list of str

mechanicalReduction
A specifier for the type of mechanical reduction.

Type str

name
The name of this actuator.

Type str

3.2.18 TransmissionJoint

class urdfpy.**TransmissionJoint**(*name*, *hardwareInterfaces*)
Bases: *urdfpy.URDFType*

A transmission joint specification.

Parameters

- **name** (*str*) – The name of this actuator.
- **hardwareInterfaces** (*list of str, optional*) – The supported hardware interfaces to the actuator.

Attributes Summary

hardwareInterfaces	The supported hardware interfaces.
name	The name of this transmission joint.

Attributes Documentation

hardwareInterfaces
The supported hardware interfaces.

Type list of str

name
The name of this transmission joint.

Type str

3.2.19 Transmission

class urdfpy.**Transmission**(*name*, *trans_type*, *joints=None*, *actuators=None*)
Bases: *urdfpy.URDFType*

An element that describes the relationship between an actuator and a joint.

Parameters

- **name** (*str*) – The name of this transmission.
- **trans_type** (*str*) – The type of this transmission.
- **joints** (list of *TransmissionJoint*) – The joints connected to this transmission.
- **actuators** (list of *Actuator*) – The actuators connected to this transmission.

Attributes Summary

actuators	The actuators the transmission is connected to.
joints	The joints the transmission is connected to.
name	The name of this transmission.
trans_type	The type of this transmission.

Attributes Documentation

actuators
The actuators the transmission is connected to.

Type *Actuator*

joints
The joints the transmission is connected to.

Type *TransmissionJoint*

name
The name of this transmission.

Type str

trans_type
The type of this transmission.

Type str

3.2.20 Joint

class urdfpy.**Joint**(*name*, *joint_type*, *parent*, *child*, *axis=None*, *origin=None*, *limit=None*, *dynamics=None*, *safety_controller=None*, *calibration=None*, *mimic=None*)

Bases: *urdfpy.URDFType*

A connection between two links.

There are several types of joints, including:

- `fixed` - a joint that cannot move.
- `prismatic` - a joint that slides along the joint axis.
- `revolute` - a hinge joint that rotates about the axis with a limited range of motion.
- `continuous` - a hinge joint that rotates about the axis with an unlimited range of motion.
- `planar` - a joint that moves in the plane orthogonal to the axis.
- `floating` - a joint that can move in 6DoF.

Parameters

- **name** (*str*) – The name of this joint.
- **parent** (*str*) – The name of the parent link of this joint.
- **child** (*str*) – The name of the child link of this joint.
- **joint_type** (*str*) – The type of the joint. Must be one of *Joint.TYPES*.
- **axis** (*(3,) float, optional*) – The axis of the joint specified in joint frame. Defaults to `[1,0,0]`.
- **origin** (*(4,4) float, optional*) – The pose of the child link with respect to the parent link's frame. The joint frame is defined to be coincident with the child link's frame, so this is also the pose of the joint frame with respect to the parent link's frame.
- **limit** (*JointLimit*, optional) – Limit for the joint. Only required for revolute and prismatic joints.
- **dynamics** (*JointDynamics*, optional) – Dynamics for the joint.
- **safety_controller** (*:class`.SafetyController`, optional*) – The safety controller for this joint.
- **calibration** (*JointCalibration*, optional) – Calibration information for the joint.
- **mimic** (*JointMimic*, optional) – Joint mimicry information.

Attributes Summary

TYPES	
axis	The joint axis in the joint frame.
calibration	The calibration for this joint.

Continued on next page

Table 21 – continued from previous page

child	The name of the child link.
dynamics	The dynamics for this joint.
joint_type	The type of this joint.
limit	The limits for this joint.
mimic	The mimic for this joint.
name	Name for this joint.
origin	The pose of child and joint frames relative to the parent link's frame.
parent	The name of the parent link.
safety_controller	The safety controller for this joint.

Methods Summary

get_child_pose([cfg])	Computes the child pose relative to a parent pose for a given configuration value.
is_valid(cfg)	Check if the provided configuration value is valid for this joint.

Attributes Documentation

TYPES = ['fixed', 'prismatic', 'revolute', 'continuous', 'floating', 'planar']

axis
The joint axis in the joint frame.

Type (3,) float

calibration
The calibration for this joint.

Type *JointCalibration*

child
The name of the child link.

Type str

dynamics
The dynamics for this joint.

Type *JointDynamics*

joint_type
The type of this joint.

Type str

limit
The limits for this joint.

Type *JointLimit*

mimic
The mimic for this joint.

Type *JointMimic*

name
Name for this joint.

Type str

origin
The pose of child and joint frames relative to the parent link's frame.

Type (4,4) float

parent
The name of the parent link.

Type str

safety_controller
The safety controller for this joint.

Type *SafetyController*

Methods Documentation

get_child_pose(*cfg=None*)
Computes the child pose relative to a parent pose for a given configuration value.

Parameters **cfg** (*float, (2,) float, (6,) float, or (4,4) float*) – The configuration values for this joint. They are interpreted based on the joint type as follows:

- `fixed` - not used.
- `prismatic` - a translation along the axis in meters.
- `revolute` - a rotation about the axis in radians.
- `continuous` - a rotation about the axis in radians.
- `planar` - the x and y translation values in the plane.
- `floating` - the xyz values followed by the rpy values, or a (4,4) matrix.

If `cfg` is `None`, then this just returns the joint pose.

Returns **pose** – The pose of the child relative to the parent.

Return type (4,4) float

is_valid(*cfg*)
Check if the provided configuration value is valid for this joint.

Parameters **cfg** (*float, (2,) float, (6,) float, or (4,4) float*) – The configuration of the joint.

Returns **is_valid** – True if the configuration is valid, and False otherwise.

Return type bool

3.2.21 Link

class urdfpy.**Link**(*name*, *inertial*, *visuals*, *collisions*)
Bases: *urdfpy.URDFType*

A link of a rigid object.

Parameters

- **name** (*str*) – The name of the link.

- **inertial** (*Inertial*, optional) – The inertial properties of the link.
- **visuals** (list of *Visual*, optional) – The visual properties of the link.
- **collsions** (list of *Collision*, optional) – The collision properties of the link.

Attributes Summary

collision_mesh	A single collision mesh for the link, specified in the link frame, or None if there isn't one.
collisions	The collision properties of this link.
inertial	Inertial properties of the link.
name	The name of this link.
visuals	The visual properties of this link.

Attributes Documentation

collision_mesh
A single collision mesh for the link, specified in the link frame, or None if there isn't one.

Type Trimesh

collisions
The collision properties of this link.

Type list of *Collision*

inertial
Inertial properties of the link.

Type *Inertial*

name
The name of this link.

Type str

visuals
The visual properties of this link.

Type list of *Visual*

3.2.22 URDF

class urdfpy.**URDF**(*name*, *links*, *joints=None*, *transmissions=None*, *materials=None*, *other_xml=None*)
Bases: *urdfpy.URDFType*

The top-level URDF specification.

The URDF encapsulates an articulated object, such as a robot or a gripper. It is made of links and joints that tie them together and define their relative motions.

Parameters

- **name** (*str*) – The name of the URDF.
- **links** (list of *Link*) – The links of the URDF.
- **joints** (list of *Joint*, optional) – The joints of the URDF.

- **transmissions** (list of *Transmission*, optional) – The transmissions of the URDF.
- **materials** (list of *Material*, optional) – The materials for the URDF.
- **other_xml** (*str, optional*) – A string containing any extra XML for extensions.

Attributes Summary

actuated_joints	The joints that are independently actuated.
base_link	The base link for the URDF.
end_links	The end links for the URDF.
joint_limit_cfgs	The lower-bound and upper-bound joint configuration maps.
joint_map	Map from joint names to the joints themselves.
joints	The links of the URDF.
link_map	Map from link names to the links themselves.
links	The links of the URDF.
material_map	Map from material names to the materials themselves.
materials	The materials of the URDF.
name	The name of the URDF.
other_xml	Any extra XML that belongs with the URDF.
transmission_map	Map from transmission names to the transmissions themselves.
transmissions	The transmissions of the URDF.

Methods Summary

animate([cfg_trajectory, loop_time, ...])	Animate the URDF through a configuration trajectory.
collision_geometry_fk([cfg, links])	Computes the poses of the URDF's collision geometries using fk.
collision_trimesh_fk([cfg, links])	Computes the poses of the URDF's collision trimeshes using fk.
link_fk([cfg, links])	Computes the poses of the URDF's links via forward kinematics.
load(file_obj)	Load a URDF from a file.
save(file_obj)	Save this URDF to a file.
show([cfg, use_collision])	Visualize the URDF in a given configuration.
visual_geometry_fk([cfg, links])	Computes the poses of the URDF's visual geometries using fk.
visual_trimesh_fk([cfg, links])	Computes the poses of the URDF's visual trimeshes using fk.

Attributes Documentation

actuated_joints

The joints that are independently actuated.

This excludes mimic joints and fixed joints.

Type list of *Joint*

base_link
The base link for the URDF.

The base link is the single link that has no parent.

Type *Link*

end_links
The end links for the URDF.

The end links are the links that have no children.

Type list of *Link*

joint_limit_cfgs
The lower-bound and upper-bound joint configuration maps.

The first map is the lower-bound map, which maps limited joints to their lower joint limits. The second map is the upper-bound map, which maps limited joints to their upper joint limits.

Type tuple of dict

joint_map
Map from joint names to the joints themselves.

This returns a copy of the joint map which cannot be edited directly. If you want to add or remove joints, use the appropriate functions.

Type dict

joints
The links of the URDF.

This returns a copy of the joints array which cannot be edited directly. If you want to add or remove joints, use the appropriate functions.

Type list of *Joint*

link_map
Map from link names to the links themselves.

This returns a copy of the link map which cannot be edited directly. If you want to add or remove links, use the appropriate functions.

Type dict

links
The links of the URDF.

This returns a copy of the links array which cannot be edited directly. If you want to add or remove links, use the appropriate functions.

Type list of *Link*

material_map
Map from material names to the materials themselves.

This returns a copy of the material map which cannot be edited directly. If you want to add or remove materials, use the appropriate functions.

Type dict

materials
The materials of the URDF.

This returns a copy of the materials array which cannot be edited directly. If you want to add or remove materials, use the appropriate functions.

Type list of *Material*

name
The name of the URDF.

Type str

other_xml
Any extra XML that belongs with the URDF.

Type str

transmission_map
Map from transmission names to the transmissions themselves.

This returns a copy of the transmission map which cannot be edited directly. If you want to add or remove transmissions, use the appropriate functions.

Type dict

transmissions
The transmissions of the URDF.

This returns a copy of the transmissions array which cannot be edited directly. If you want to add or remove transmissions, use the appropriate functions.

Type list of *Transmission*

Methods Documentation

animate(*cfg_trajectory=None*, *loop_time=3.0*, *use_collision=False*)
Animate the URDF through a configuration trajectory.

Parameters

- **cfg_trajectory** (*dict*) – A map from joints or joint names to lists of configuration values for each joint along the trajectory. If not specified, all joints will articulate from limit to limit. The trajectory steps are assumed to be equally spaced out in time.
- **loop_time** (*float*) – The time to loop the animation for, in seconds. The trajectory will play fowards and backwards during this time, ending at the inital configuration.
- **use_collision** (*bool*) – If True, the collision geometry is visualized instead of the visual geometry.

Examples

You can run this without specifying a `cfg_trajectory` to view the full articulation of the URDF

```
>>> robot = URDF.load('ur5.urdf')
>>> robot.animate()
```

```
>>> ct = {'shoulder_pan_joint': [0.0, 2 * np.pi]}
>>> robot.animate(cfg_trajectory=ct)
```

```
>>> ct = {
...     'shoulder_pan_joint' : [-np.pi / 4, np.pi / 4],
...     'shoulder_lift_joint' : [0.0, -np.pi / 2.0],
...     'elbow_joint' : [0.0, np.pi / 2.0]
... }
>>> robot.animate(cfg_trajectory=ct)
```

collision_geometry_fk(*cfg=None*, *links=None*)
Computes the poses of the URDF's collision geometries using fk.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **links** (list of str or list of *Link*) – The links or names of links to perform forward kinematics on. Only geometries from these links will be in the returned map. If not specified, all links are returned.

Returns **fk** – A map from *Geometry* objects that are part of the collision elements of the specified links to the 4x4 homogenous transform matrices that position them relative to the base link's frame.

Return type dict

collision_trimesh_fk(*cfg=None*, *links=None*)
Computes the poses of the URDF's collision trimeshes using fk.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **links** (list of str or list of *Link*) – The links or names of links to perform forward kinematics on. Only trimeshes from these links will be in the returned map. If not specified, all links are returned.

Returns **fk** – A map from `Trimesh` objects that are part of the collision geometry of the specified links to the 4x4 homogenous transform matrices that position them relative to the base link's frame.

Return type dict

link_fk(*cfg=None*, *links=None*)
Computes the poses of the URDF's links via forward kinematics.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **links** (list of str or list of *Link*) – The links or names of links to perform forward kinematics on. Only these links will be in the returned map. If not specified, all links are returned.

Returns **fk** – A map from links to 4x4 homogenous transform matrices that position them relative to the base link's frame.

Return type dict

static load(*file_obj*)
Load a URDF from a file.

Parameters **file_obj** (*str or file-like object*) – The file to load the URDF from. Should be the path to the `.urdf` XML file. Any paths in the URDF should be specified as relative paths to the `.urdf` file instead of as ROS resources.

Returns **urdf** – The parsed URDF.

Return type *URDF*

save(*file_obj*)
Save this URDF to a file.

Parameters **file_obj** (*str or file-like object*) – The file to save the URDF to. Should be the path to the `.urdf` XML file. Any paths in the URDF should be specified as relative paths to the `.urdf` file instead of as ROS resources.

Returns **urdf** – The parsed URDF.

Return type *URDF*

show(*cfg=None*, *use_collision=False*)
Visualize the URDF in a given configuration.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **use_collision** (*bool*) – If True, the collision geometry is visualized instead of the visual geometry.

visual_geometry_fk(*cfg=None*, *links=None*)
Computes the poses of the URDF's visual geometries using fk.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **links** (list of str or list of *Link*) – The links or names of links to perform forward kinematics on. Only geometries from these links will be in the returned map. If not specified, all links are returned.

Returns **fk** – A map from *Geometry* objects that are part of the visual elements of the specified links to the 4x4 homogenous transform matrices that position them relative to the base link's frame.

Return type dict

visual_trimesh_fk(*cfg=None*, *links=None*)
Computes the poses of the URDF's visual trimeshes using fk.

Parameters

- **cfg** (*dict*) – A map from joints or joint names to configuration values for each joint. If not specified, all joints are assumed to be in their default configurations.
- **links** (list of str or list of *Link*) – The links or names of links to perform forward kinematics on. Only trimeshes from these links will be in the returned map. If not specified, all links are returned.

Returns **fk** – A map from `Trimesh` objects that are part of the visual geometry of the specified links to the 4x4 homogenous transform matrices that position them relative to the base link's frame.

Return type dict

CHAPTER 4

Development Guide

Read this guide before doing development in `urdfpy`.

4.1 Setting Up

To set up the tools you'll need for developing, you'll need to install `urdfpy` in development mode. Start by installing the development dependencies:

```
git clone https://github.com/mmatl/urdfpy.git
cd urdfpy
pip install -e .[dev,docs]
```

Next, install the git pre-commit hooks with the following command:

```
pre-commit install
```

This step is crucial. It will install a pre-commit git hook that runs the `flake8` style checker on your code before allowing you to commit. The style checker isn't just an annoyance – it does a lot more than just check for trailing whitespace! It can identify missing or undefined variable names, unused imports, and a whole host of other potential problems in your code.

4.2 Running Code Style Checks

To run the style checker on all files under version control, simply run the following command:

```
pre-commit run --all-files
```

If you'd like to run the style checker more selectively, you can run the `flake8` command directly on a directory (recursively) or on an individual file:

```
flake8 --ignore=E231,W504 /path/to/files
```

For more information, please see the flake8 documentation.

4.3 Running Tests

This project uses pytest, the standard Python testing framework. Their website has tons of useful details, but here are the basics.

To run the testing suite, simply navigate to the top-level folder in `urdfpy` and run the following command:

```
pytest -v tests
```

You should see the testing suite run. There are a few useful command line options that are good to know:

- `-s` - Shows the output of `stdout`. By default, this output is masked.
- `--pdb` - Instead of crashing, opens a debugger at the first fault.
- `--lf` - Instead of running all tests, just run the ones that failed last.
- `--trace` - Open a debugger at the start of each test.

You can see all of the other command-line options here.

By default, `pytest` will look in the `tests` folder recursively. It will run any function that starts with `test_` in any file that starts with `test_`. You can run `pytest` on a directory or on a particular file by specifying the file path:

```
pytest -v tests/unit/utils/test_transforms.py
```

You can also test a single function in a file:

```
pytest -v tests/unit/utils/test_images.py::test_rgbd_image
```

4.4 Generating Code Coverage Reports

To generate code coverage documentation, re-run the tests with the `--cov` option set:

```
pytest --cov=urdfpy tests
```

This will dump a code coverage file. To view it in a nice HTML document, run the following command:

```
coverage html
```

The generated HTML homepage will be stored at `htmlcov/index.html`.

4.5 Building Documentation

To build `urdfpy`'s documentation, go to the `docs` directory and run `make` with the appropriate target. For example,

```
cd docs/
make html
```

will generate HTML-based docs, which are probably the easiest to read. The resulting index page is at `docs/build/html/index.html`. If the docs get stale, just run `make clean` to remove all build files.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Index

M

N

O

P

R

S

T

U

V

X