

---

# **yourdfpy Documentation**

***Release 0.0.52.post1.dev2+gf75868f***

**Clemens Eppner**

**Jan 20, 2023**



# CONTENTS

<b>1</b>	<b>Contents</b>	<b>3</b>
<b>2</b>	<b>Indices and tables</b>	<b>25</b>
	<b>Python Module Index</b>	<b>27</b>
	<b>Index</b>	<b>29</b>



Yet anOther URDF parser for Python. Yup, it's another one. Deal with it.

Yourdfpy is a simpler and easier-to-use library for loading, manipulating, validating, saving, and visualizing URDF files.



## CONTENTS

[Build Status](#) [Documentation Status](#) [Coverage Status](#) [PyPI version](#)

## 1.1 yourdfpy

Yet anOther URDF parser for Python. Yup, it's another one. Deal with it.

Yourdfpy is a simpler and easier-to-use library for loading, manipulating, validating, saving, and visualizing URDF files.

### 1.1.1 Installation

You can install yourdfpy directly from pip:

```
pip install yourdfpy
```

### 1.1.2 Visualization

Once installed, you can visualize a URDF model from the command line:

```
yourdfpy ./my_description/urdf/robot.urdf
```

You can use the following keyboard shortcuts to inspect your model:

- a: Toggle rendered XYZ/RGB axis markers (off, world frame, every frame)
- w: Toggle wireframe mode (good for looking inside meshes, off by default)
- c: Toggle back face culling (on by default but in wireframe mode it is sometimes useful to see the back sides)

### 1.1.3 But why another one!?

Why are you wasting not only your but also our time? you might ask. Fair point. There are already [urdfpy](#) and [urdf\\_parser\\_py](#) that deal with URDFs. Unfortunately, none of these solutions allow customizable URDF parsing that is fully independent of validation and mesh loading. Dealing with filenames, outdated dependencies, open bug reports, and limited flexibility when it comes to serialization are other disadvantages. As shown in the table below, **yourdfpy** is the most robust one when it comes to loading URDFs in the wild.



Example

URDFs

```
|| urdfpy | urdf_parser_py | yourdfpy || _____:
|: _____: |: _____: |: _____: || Decouple pars-
ing from validation || |:heavy_check_mark: || Decouple parsing from loading meshes || |:heavy_check_mark: |
:heavy_check_mark: || Visualize URDF |:heavy_check_mark: || |:heavy_check_mark: || Forward Kinematics |
:heavy_check_mark: || |:heavy_check_mark: || Robustness test: loading 12 URDF files from here | 4/12 | 6/12 | 12/12
|| Avg. loading time per file (w/ mesh loading) | 480 ms || 370 ms || (w/o mesh loading) || 3.2 ms | 6.2 ms || Test on
4 URDF files on which urdfpy succeeds | 347.5 ms || 203 ms || Test on 6 URDF files on which urdf_parser_py
succeeds || 2.6 ms | 3.8 ms |
```

```
robot_assets = ['robot-assets/urdfs/robots/barret_hand/bhand_model.URDF', 'robot-assets/
↳ urdfs/robots/robotiq_gripper/robotiq_arg85_description.URDF', 'robot-assets/urdfs/
↳ robots/anymal/anymal.urdf', 'robot-assets/urdfs/robots/franka_panda/panda.urdf',
↳ 'robot-assets/urdfs/robots/ginger_robot/gingerurdf.urdf', 'robot-assets/urdfs/robots/
↳ halodi/eve_r3.urdf', 'robot-assets/urdfs/robots/kinova/kinova.urdf', 'robot-assets/
↳ urdfs/robots/kuka_iiwa/model.urdf', 'robot-assets/urdfs/robots/pr2/pr2.urdf', 'robot-
↳ assets/urdfs/robots/ur10/ur10_robot.urdf', 'robot-assets/urdfs/robots/ur5/ur5_gripper.
↳ urdf', 'robot-assets/urdfs/robots/yumi/yumi.urdf']
```

```
import urdfpy
import urdf_parser_py
import yourdfpy
```

```
from functools import partial
```

```
def load_urdfs(fnames, load_fn):
    results = {fname: None for fname in fnames}
    for fname in fnames:
        try:
            x = load_fn(fname)
            results[fname] = x
        except:
            print("Problems loading: ", fname)
            pass
    print(sum([1 for x, y in results.items() if y is not None]), "/", len(fnames))
    return results
```

```
# parsing success rate
```

```
load_urdfs(robot_assets, urdfpy.URDF.load)
load_urdfs(robot_assets, urdf_parser_py.urdf.URDF.load)
load_urdfs(robot_assets, yourdfpy.URDF.load)
```

```
# parsing times
```

```
%timeit load_urdfs(robot_assets, urdfpy.URDF.load)
%timeit load_urdfs(robot_assets, urdf_parser_py.urdf.URDF.load)
%timeit load_urdfs(robot_assets, yourdfpy.URDF.load)
%timeit load_urdfs(robot_assets, partial(yourdfpy.URDF.load, load_meshes=False, build_
↳ scene_graph=False))
```

(continues on next page)



(continued from previous page)

```
# fairer comparison with yourdfpy
urdfpy_fnames = [x for x, y in load_urdfs(robot_assets, urdfpy.URDF.load).items() if y
↳ is not None]
%timeit load_urdfs(urdfpy_fnames, yourdfpy.URDF.load)

# fairer comparison with urdf_parser_py
urdfparser_fnames = [x for x, y in load_urdfs(robot_assets, urdf_parser_py.urdf.URDF.
↳ from_xml_file).items() if y is not None]
%timeit load_urdfs(urdfparser_fnames, functools.partial(yourdfpy.URDF.load, load_
↳ meshes=False, build_scene_graph=False))
```

## 1.2 License

The MIT License (MIT)

Copyright (c) 2021 Clemens Eppner

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the “Software”), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## 1.3 Contributors

- Clemens Eppner

## 1.4 Changelog

### 1.4.1 Version 0.0.53

- Fix NumPy float issue #46: <https://github.com/clemense/yourdfpy/issues/45>

### 1.4.2 Version 0.0.52

- `parse_mass()` returns float instead of str
- default parsed mass is 0.0 instead of 1.0 (see <http://wiki.ros.org/urdf/XML/link>)
- Update `trimesh` dependency to `trimesh[easy]` (to support loading Collada meshes)
- Won't crash when loading model with joints that mimic unactuated/fixed joints.

### 1.4.3 Version 0.0.51

- Fix path separator issues in Windows [Issue 27](#)

### 1.4.4 Version 0.0.50

- A nothingburger

### 1.4.5 Version 0.0.49

- Fix single link URDF trimesh scene bug [PR26](#)

### 1.4.6 Version 0.0.48

- Implement `-c/--configuration` argument for `yourdfpy`
- Add `--animate` flag to `yourdfpy`

### 1.4.7 Version 0.0.47

- Bugfix: Parsing box dimensions
- Change to `trimesh.primitives.*` for geometric primitives (instead of `trimesh.creation.*`)

### 1.4.8 Version 0.0.46

- Bugfix: Named material with color wouldn't be applied

### 1.4.9 Version 0.0.45

- Upgrade to `trimesh` version 3.11.2
- Add `__eq__` operator to URDF based on equality of individual elements (order-invariant) [PR18](#)
- Add material information [PR15](#)
- Improve mesh filename search [PR14](#)

#### 1.4.10 Version 0.0.44

- Parse and write name attribute of material element
- Apply colors to mesh if available
- Handle empty scene exception (in case of URDF without meshes)

#### 1.4.11 Version 0.0.43

- Skip material loading for collision geometries

#### 1.4.12 Version 0.0.42

- Fix bug when updating robots with mimic joints

#### 1.4.13 Version 0.0.41

- yourdfpy.Sphere visible

#### 1.4.14 Version 0.0.40

- Add 'file\_path', 'file\_name', and 'file\_element' to trimesh geometry's metadata in case loaded OBJ file contains multiple parts
- Load collision geometry for viz

#### 1.4.15 Version 0.0.39

- Fix mimic joint issue

#### 1.4.16 Version 0.0.38

- Change namespace of filename handlers

#### 1.4.17 Version 0.0.37

- Use visual/collision name property as geometry name for scene graph
- Write visual/collision name property to URDF

#### 1.4.18 Version 0.0.36

- Fix validation of JointLimit
- Add Dynamics to `init.py`

#### 1.4.19 Version 0.0.35

- Add `force_single_geometry_per_link` feature: similar to `collision_mesh` in urdfpy; will concatenate all meshes in a single link and only create one node in the scene graph. This is the new default for loading the collision scene.

#### 1.4.20 Version 0.0.34

- Fix missing Collision exposure in `init.py`
- Add `force_collision_mesh`

#### 1.4.21 Version 0.0.33

- Add `force_mesh` to constructor; allows loading mesh files as single meshes instead of turning them into graphs (since trimesh can't deal with meshes with multiple textures)

#### 1.4.22 Version 0.0.32

- Fix continuous joint during forward kinematics
- Introduce DOF indices for actuated joints (to handle planar and floating types)

#### 1.4.23 Version 0.0.31

- Add `num_dofs`
- Add `zero_cfg` property
- Change function `get_default_cfg` to property `center_cfg`
- Initial configuration is now the zero configuration not the center configuration (as previously)

#### 1.4.24 Version 0.0.30

#### 1.4.25 Version 0.0.29

#### 1.4.26 Version 0.0.28

#### 1.4.27 Version 0.0.27

- Bugfix in travis deployment pipeline

### 1.4.28 Version 0.0.26

- Bugfix: rename `generate_scene_graph` parameter
- Bugfix of bugfix of previous version, which introduced a new bug
- Bugfix: root URDF result of `split_along_joints` (scene was not in sync with model)
- Add params to `split_along_joints`
- Bugfix: `parse_inertia` resulted in wrong matrix dtype

### 1.4.29 Version 0.0.25

- Bugfix: `get_default_cfg` returns flattened array

### 1.4.30 Version 0.0.24

- Added `pytest`s
- Separate visual and collision scene
- Rename constructor's parameter `create_scene_graph` to `build_scene_graph`
- Added ROS validation rules
- Rename `update_trimesh_scene` to `update_cfg`, change arguments
- Add `get_transform` function
- Rename `get_default_configuration` to `get_default_cfg`
- Proper handling of mimic joints
- New members for `actuated_joints`
- New `base_link` property

### 1.4.31 Version 0.0.23

- The Great YOURDFPY Steering Committee (G.Y.S.C.) decides to jump as many version numbers ahead as needed to pass urdfpy

### 1.4.32 Version 0.0.14

- The Great YOURDFPY Steering Committee (G.Y.S.C.) gives up on using only version numbers that are prime

### 1.4.33 Version 0.0.13

- Adding images. For the Github crowd.

### 1.4.34 Version 0.0.11

- These numbers are going up quickly.

### 1.4.35 Version 0.0.7

- Wow. This was quite the evening.

### 1.4.36 Version 0.0.5

- The Great YOURDFPY Steering Committee (G.Y.S.C.) decides to only use version numbers that are prime

### 1.4.37 Version 0.0.3

- A version few remember and many ignored

### 1.4.38 Version 0.0.1

- A version nobody remembers

## 1.5 yourdfpy

### 1.5.1 yourdfpy package

#### Submodules

#### yourdfpy.urdf module

```
class yourdfpy.urdf.Actuator(name: str, mechanical_reduction: Union[float, NoneType] = None,
                              hardware_interfaces: List[str] = <factory>)
```

Bases: `object`

**hardware\_interfaces:** `List[str]`

**mechanical\_reduction:** `float | None = None`

**name:** `str`

```
class yourdfpy.urdf.Box(size: numpy.ndarray)
```

Bases: `object`

**size:** `ndarray`

```
class yourdfpy.urdf.Calibration(rising: float | NoneType = None, falling: float | NoneType = None)
    Bases: object
    falling: float | None = None
    rising: float | None = None

class yourdfpy.urdf.Collision(name: str, origin: numpy.ndarray | NoneType = None, geometry:
    yourdfpy.urdf.Geometry = None)
    Bases: object
    geometry: Geometry = None
    name: str
    origin: ndarray | None = None

class yourdfpy.urdf.Color(rgba: numpy.ndarray)
    Bases: object
    rgba: ndarray

class yourdfpy.urdf.Cylinder(radius: float, length: float)
    Bases: object
    length: float
    radius: float

class yourdfpy.urdf.Dynamics(damping: float | NoneType = None, friction: float | NoneType = None)
    Bases: object
    damping: float | None = None
    friction: float | None = None

class yourdfpy.urdf.Geometry(box: yourdfpy.urdf.Box | NoneType = None, cylinder: yourdfpy.urdf.Cylinder |
    NoneType = None, sphere: yourdfpy.urdf.Sphere | NoneType = None, mesh:
    yourdfpy.urdf.Mesh | NoneType = None)
    Bases: object
    box: Box | None = None
    cylinder: Cylinder | None = None
    mesh: Mesh | None = None
    sphere: Sphere | None = None

class yourdfpy.urdf.Inertial(origin: numpy.ndarray | NoneType = None, mass: float | NoneType = None,
    inertia: numpy.ndarray | NoneType = None)
    Bases: object
    inertia: ndarray | None = None
    mass: float | None = None
    origin: ndarray | None = None
```

```
class yourdfpy.urdf.Joint(name: str, type: str = None, parent: str = None, child: str = None, origin:
    numpy.ndarray = None, axis: numpy.ndarray = None, dynamics:
    yourdfpy.urdf.Dynamics | NoneType = None, limit: yourdfpy.urdf.Limit | NoneType
    = None, mimic: yourdfpy.urdf.Mimic | NoneType = None, calibration:
    yourdfpy.urdf.Calibration | NoneType = None, safety_controller:
    yourdfpy.urdf.SafetyController | NoneType = None)
```

Bases: `object`

**axis:** `ndarray` = `None`

**calibration:** `Calibration` | `None` = `None`

**child:** `str` = `None`

**dynamics:** `Dynamics` | `None` = `None`

**limit:** `Limit` | `None` = `None`

**mimic:** `Mimic` | `None` = `None`

**name:** `str`

**origin:** `ndarray` = `None`

**parent:** `str` = `None`

**safety\_controller:** `SafetyController` | `None` = `None`

**type:** `str` = `None`

```
class yourdfpy.urdf.Limit(effort: float | NoneType = None, velocity: float | NoneType = None, lower: float |
    NoneType = None, upper: float | NoneType = None)
```

Bases: `object`

**effort:** `float` | `None` = `None`

**lower:** `float` | `None` = `None`

**upper:** `float` | `None` = `None`

**velocity:** `float` | `None` = `None`

```
class yourdfpy.urdf.Link(name: str, inertial: Union[yourdfpy.urdf.Inertial, NoneType] = None, visuals:
    List[yourdfpy.urdf.Visual] = <factory>, collisions: List[yourdfpy.urdf.Collision] =
    <factory>)
```

Bases: `object`

**collisions:** `List[Collision]`

**inertial:** `Inertial` | `None` = `None`

**name:** `str`

**visuals:** `List[Visual]`

```
class yourdfpy.urdf.Material(name: str | NoneType = None, color: yourdfpy.urdf.Color | NoneType = None,
    texture: yourdfpy.urdf.Texture | NoneType = None)
```

Bases: `object`



```
    color: Color | None = None

    name: str | None = None

    texture: Texture | None = None

class yourdfpy.urdf.Mesh(filename: str, scale: float | numpy.ndarray | NoneType = None)
    Bases: object

    filename: str

    scale: float | ndarray | None = None

class yourdfpy.urdf.Mimic(joint: str, multiplier: float | NoneType = None, offset: float | NoneType = None)
    Bases: object

    joint: str

    multiplier: float | None = None

    offset: float | None = None

class yourdfpy.urdf.Robot(name: str, links: List[yourdfpy.urdf.Link] = <factory>, joints:
    List[yourdfpy.urdf.Joint] = <factory>, materials: List[yourdfpy.urdf.Material] =
    <factory>, transmission: List[str] = <factory>, gazebo: List[str] = <factory>)

    Bases: object

    gazebo: List[str]

    joints: List[Joint]

    links: List[Link]

    materials: List[Material]

    name: str

    transmission: List[str]

class yourdfpy.urdf.SafetyController(soft_lower_limit: float | NoneType = None, soft_upper_limit: float |
    NoneType = None, k_position: float | NoneType = None, k_velocity:
    float | NoneType = None)

    Bases: object

    k_position: float | None = None

    k_velocity: float | None = None

    soft_lower_limit: float | None = None

    soft_upper_limit: float | None = None

class yourdfpy.urdf.Sphere(radius: float)
    Bases: object

    radius: float

class yourdfpy.urdf.Texture(filename: str)
    Bases: object
```

filename: `str`

```
class yourdfpy.urdf.Transmission(name: str, type: Union[str, NoneType] = None, joints:
    List[yourdfpy.urdf.TransmissionJoint] = <factory>, actuators:
    List[yourdfpy.urdf.Actuator] = <factory>)
```

Bases: `object`

actuators: `List[Actuator]`

joints: `List[TransmissionJoint]`

name: `str`

type: `str | None = None`

```
class yourdfpy.urdf.TransmissionJoint(name: str, hardware_interfaces: List[str] = <factory>)
```

Bases: `object`

hardware\_interfaces: `List[str]`

name: `str`

```
class yourdfpy.urdf.URDF(robot: Robot | None = None, build_scene_graph: bool = True,
    build_collision_scene_graph: bool = False, load_mesher: bool = True,
    load_collision_mesher: bool = False, filename_handler=None, mesh_dir: str = "",
    force_mesh: bool = False, force_collision_mesh: bool = True)
```

Bases: `object`

**property actuated\_dof\_indices**

List of DOF indices per actuated joint. Can be used to reference configuration.

**Returns**

List of DOF indices per actuated joint.

**Return type**

`list[list[int]]`

**property actuated\_joint\_indices**

List of indices of all joints that are actuated, i.e., not of type mimic or fixed.

**Returns**

List of indices of actuated joints.

**Return type**

`list[int]`

**property actuated\_joint\_names**

List of names of actuated joints. This excludes mimic and fixed joints.

**Returns**

List of names of actuated joints of the URDF model.

**Return type**

`list[str]`

**property actuated\_joints**

List of actuated joints. This excludes mimic and fixed joints.

**Returns**

List of actuated joints of the URDF model.

**Return type**`list[Joint]`**property base\_link**

Name of URDF base/root link.

**Returns**

Name of base link of URDF model.

**Return type**`str`**property center\_cfg**

Return center configuration of URDF model by using the average of each joint's limits if present, otherwise zero.

**Returns**

Default configuration of URDF model.

**Return type**`(n), float`**property cfg**

Current configuration.

**Returns**

Current configuration of URDF model.

**Return type**`np.ndarray`**clear\_errors()**

Clear the validation error log.

**property collision\_scene: Scene**

A scene object representing the <collision> elements of the URDF model

**Returns**

A trimesh scene object.

**Return type**`trimesh.Scene`**contains**(*key*, *value*, *element=None*) → `bool`

Checks recursively whether the URDF tree contains the provided key-value pair.

**Parameters**

- **key** (`str`) – A key.
- **value** (`str`) – A value.
- **element** (`etree.Element`, *optional*) – The XML element from which to start the recursive search. None means URDF root. Defaults to None.

**Returns**

Whether the key-value pair was found.

**Return type**`bool`

**property errors:** `list`

A list with validation errors.

**Returns**

A list of validation errors.

**Return type**

`list`

**get\_transform**(*frame\_to*, *frame\_from*=None, *collision\_geometry*=False)

Get the transform from one frame to another.

**Parameters**

- **frame\_to** (`str`) – Node name.
- **frame\_from** (`str`, *optional*) – Node name. If None it will be set to self.base\_frame. Defaults to None.
- **collision\_geometry** (`bool`, *optional*) – Whether to use the collision geometry scene graph (instead of the visual geometry). Defaults to False.

**Raises**

**ValueError** – Raised if scene graph wasn't constructed during initialization.

**Returns**

Homogeneous transformation matrix

**Return type**

(4, 4) `float`

**property joint\_map:** `dict`

A dictionary mapping joint names to joint objects.

**Returns**

Mapping from joint name (str) to Joint.

**Return type**

`dict`

**property joint\_names**

List of joint names.

**Returns**

List of joint names of the URDF model.

**Return type**

`list[str]`

**property link\_map:** `dict`

A dictionary mapping link names to link objects.

**Returns**

Mapping from link name (str) to Link.

**Return type**

`dict`

**static load**(*fname\_or\_file*, *\*\*kwargs*)

Load URDF file from filename or file object.

**Parameters**

- **fname\_or\_file** (*str* or *file object*) – A filename or file object, file-like object, stream representing the URDF file.
- **\*\*build\_scene\_graph** (*bool*, *optional*) – Whether to build a scene graph to enable transformation queries and forward kinematics. Defaults to True.
- **\*\*build\_collision\_scene\_graph** (*bool*, *optional*) – Whether to build a scene graph for <collision> elements. Defaults to False.
- **\*\*load\_meshes** (*bool*, *optional*) – Whether to load the meshes referenced in the <mesh> elements. Defaults to True.
- **\*\*load\_collision\_meshes** (*bool*, *optional*) – Whether to load the collision meshes referenced in the <mesh> elements. Defaults to False.
- **\*\*filename\_handler** (*[type]*, *optional*) – Any function *f*(in: str) -> str, that maps filenames in the URDF to actual resources. Can be used to customize treatment of *package://* directives or relative/absolute filenames. Defaults to None.
- **\*\*mesh\_dir** (*str*, *optional*) – A root directory used for loading meshes. Defaults to “.”.
- **\*\*force\_mesh** (*bool*, *optional*) – Each loaded geometry will be concatenated into a single one (instead of being turned into a graph; in case the underlying file contains multiple geometries). This might lose texture information but the resulting scene graph will be smaller. Defaults to False.
- **\*\*force\_collision\_mesh** (*bool*, *optional*) – Same as *force\_mesh*, but for collision scene. Defaults to True.

**Raises**

**ValueError** – If filename does not exist.

**Returns**

URDF model.

**Return type**

yourdfpy.URDF

**property num\_actuated\_joints**

Number of actuated joints.

**Returns**

Number of actuated joints.

**Return type**

int

**property num\_dofs**

Number of degrees of freedom of actuated joints. Depending on the type of the joint, the number of DOFs might vary.

**Returns**

Degrees of freedom.

**Return type**

int

**property scene: Scene**

A scene object representing the URDF model.

**Returns**

A trimesh scene object.

**Return type**

trimesh.Scene

**show**(*collision\_geometry=False, callback=None*)

Open a simpler viewer displaying the URDF model.

**Parameters**

**collision\_geometry** (*bool, optional*) – Whether to display the <collision> or <visual> elements. Defaults to False.

**split\_along\_joints**(*joint\_type='floating', \*\*kwargs*)

Split URDF model along a particular joint type. The result is a set of URDF models which together compose the original URDF.

**Parameters**

- **joint\_type** (*str, or list[str], optional*) – Type of joint to use for splitting. Defaults to “floating”.
- **\*\*kwargs** – Arguments delegated to URDF constructor of new URDF models.

**Returns**

A list of tuples (np.ndarray, yourdfpy.URDF) whereas each homogeneous 4x4 matrix describes the root transformation of the respective URDF model w.r.t. the original URDF.

**Return type**

list[(np.ndarray, yourdfpy.URDF)]

**update\_cfg**(*configuration*)

Update joint configuration of URDF; does forward kinematics.

**Parameters**

**configuration** (*dict, list[float], tuple[float] or np.ndarray*) – A mapping from joints or joint names to configuration values, or a list containing a value for each actuated joint.

**Raises**

- **ValueError** – Raised if dimensionality of configuration does not match number of actuated joints of URDF model.
- **TypeError** – Raised if configuration is neither a dict, list, tuple or np.ndarray.

**validate**(*validation\_fn=None*) → bool

Validate URDF model.

**Parameters**

**validation\_fn** (*function, optional*) – A function f(list[yourdfpy.URDFError]) → bool. None uses the strict handler (any error leads to False). Defaults to None.

**Returns**

Whether the model is valid.

**Return type**

bool

**validate\_filenames**()

**write\_xml**()

Write URDF model to an XML element hierarchy.

**Returns**

XML data.

**Return type**

etree.ElementTree

**write\_xml\_file(*fname*)**

Write URDF model to an xml file.

**Parameters****fname** (*str*) – Filename of the file to be written. Usually ends in *.urdf*.**write\_xml\_string(\*\**kwargs*)**

Write URDF model to a string.

**Returns**

String of the xml representation of the URDF model.

**Return type**

str

**property zero\_cfg**

Return the zero configuration.

**Returns**

The zero configuration.

**Return type**

np.ndarray

**exception yourdfpy.urdf.URDFAttributeValueError(*msg*)**Bases: [URDFError](#)

Raised when attribute value is not contained in the set of allowed values.

**exception yourdfpy.urdf.URDFBrokenRefError(*msg*)**Bases: [URDFError](#)

Raised when a referenced object is not found in the scope.

**exception yourdfpy.urdf.URDFError(*msg*)**Bases: [Exception](#)

General URDF exception.

**exception yourdfpy.urdf.URDFIncompleteError(*msg*)**Bases: [URDFError](#)

Raised when needed data for an object isn't there.

**exception yourdfpy.urdf.URDFMalformedError(*msg*)**Bases: [URDFError](#)

Raised when data is found to be corrupted in some way.

**exception yourdfpy.urdf.URDFSaveValidationError(*msg*)**Bases: [URDFError](#)

Raised when XML validation fails when saving.

**exception yourdfpy.urdf.URDFUnsupportedError(*msg*)**Bases: [URDFError](#)

Raised when some unexpectedly unsupported feature is found.

```
class yourdfpy.urdf.Visual(name: str | NoneType = None, origin: numpy.ndarray | NoneType = None,
                           geometry: yourdfpy.urdf.Geometry | NoneType = None, material:
                           yourdfpy.urdf.Material | NoneType = None)
```

Bases: `object`

**geometry:** `Geometry` | `None` = `None`

**material:** `Material` | `None` = `None`

**name:** `str` | `None` = `None`

**origin:** `ndarray` | `None` = `None`

```
yourdfpy.urdf.apply_visual_color(geom: Trimesh, visual: Visual, material_map: Dict[str, Material]) →
                                None
```

Apply the color of the visual material to the mesh.

#### Parameters

- **geom** – Trimesh to color.
- **visual** – Visual description from XML.
- **material\_map** – Dictionary mapping material names to their definitions.

```
yourdfpy.urdf.filename_handler_absolute2relative(fname, dir)
```

A filename handler that turns an absolute file name into a relative one.

#### Parameters

- **fname** (`str`) – A file name.
- **dir** (`str`) – A directory.

#### Returns

The file name relative to the directory.

#### Return type

`str`

```
yourdfpy.urdf.filename_handler_add_prefix(fname, prefix)
```

A filename handler that adds a prefix.

#### Parameters

- **fname** (`str`) – A file name.
- **prefix** (`str`) – A prefix.

#### Returns

Prefix plus file name.

#### Return type

`str`

```
yourdfpy.urdf.filename_handler_ignore_directive(fname)
```

A filename handler that removes anything before (and including) `“:”`.

#### Parameters

**fname** (`str`) – A file name.

#### Returns

The file name without the prefix.



**Return type**`str``yourdfpy.urdf.filename_handler_ignore_directive_package(fname)`

A filename handler that removes the ‘package://’ directive and the package it refers to. It subsequently calls `filename_handler_ignore_directive`, i.e., it removes any other directive.

**Parameters**

**fname** (`str`) – A file name.

**Returns**

The file name without ‘package://’ and the package name.

**Return type**`str``yourdfpy.urdf.filename_handler_magic(fname, dir)`

A magic filename handler.

**Parameters**

- **fname** (`str`) – A file name.
- **dir** (`str`) – A directory.

**Returns**

The file name that exists or the input if nothing is found.

**Return type**`str``yourdfpy.urdf.filename_handler_meta(fname, filename_handlers)`

A filename handler that calls other filename handlers until the resulting file name points to an existing file.

**Parameters**

- **fname** (`str`) – A file name.
- **filename\_handlers** (`list(fn)`) – A list of function pointers to filename handlers.

**Returns**

The resolved file name that points to an existing file or the input if none of the files exists.

**Return type**`str``yourdfpy.urdf.filename_handler_null(fname)`

A lazy filename handler that simply returns its input.

**Parameters**

**fname** (`str`) – A file name.

**Returns**

Same file name.

**Return type**`str``yourdfpy.urdf.filename_handler_relative(fname, dir)`

A filename handler that joins a file name with a directory.

**Parameters**

- **fname** (`str`) – A file name.

- **dir** (*str*) – A directory.

**Returns**

The directory joined with the file name.

**Return type**

*str*

`yourdfpy.urdf.filename_handler_relative_to_urdf_file(fname, urdf_fname)`

`yourdfpy.urdf.filename_handler_relative_to_urdf_file_recursive(fname, urdf_fname, level=0)`

`yourdfpy.urdf.validation_handler_strict(errors)`

A validation handler that does not allow any errors.

**Parameters**

**errors** (*list* [*yourdfpy.URDFError*]) – List of errors.

**Returns**

Whether any errors were found.

**Return type**

*bool*

## yourdfpy.viz module

Script for visualizing a robot from a URDF.

`yourdfpy.viz.generate_joint_limit_trajectory(urdf_model, loop_time)`

Generate a trajectory for all actuated joints that interpolates between joint limits. For continuous joint interpolate between  $[0, 2 * \pi]$ .

**Parameters**

- **urdf\_model** (*yourdfpy.URDF*) – `_description_`
- **loop\_time** (*float*) – Time in seconds to loop through the trajectory.

**Returns**

A dictionary over all actuated joints with list of configuration values.

**Return type**

*dict*

`yourdfpy.viz.main(args)`

Wrapper allowing string arguments in a CLI fashion.

**Parameters**

**args** (*List* [*str*]) – command line parameters as list of strings (for example `[ "--verbose", "42" ]`).

`yourdfpy.viz.parse_args(args)`

Parse command line parameters

**Parameters**

**args** (*List* [*str*]) – command line parameters as list of strings (for example `[ "--help" ]`).

**Returns**

command line parameters namespace

**Return type**

*argparse.Namespace*

`yourdfpy.viz.run()`

Calls `main()` passing the CLI arguments extracted from `sys.argv`.

This function can be used as entry point to create console scripts with `setuptools`.

`yourdfpy.viz.setup_logging(loglevel)`

Setup basic logging.

**Parameters**

**loglevel** (*int*) – minimum loglevel for emitting messages

`yourdfpy.viz.viewer_callback(scene, urdf_model, trajectory, loop_time)`

## Module contents



## INDICES AND TABLES

- `genindex`
- `modindex`
- `search`



## PYTHON MODULE INDEX

### y

`yourdfpy`, [23](#)  
`yourdfpy.urdf`, [10](#)  
`yourdfpy.viz`, [22](#)





## A

actuated\_dof\_indices (*yourdfpy.urdf.URDF* property), 14  
 actuated\_joint\_indices (*yourdfpy.urdf.URDF* property), 14  
 actuated\_joint\_names (*yourdfpy.urdf.URDF* property), 14  
 actuated\_joints (*yourdfpy.urdf.URDF* property), 14  
 Actuator (class in *yourdfpy.urdf*), 10  
 actuators (*yourdfpy.urdf.Transmission* attribute), 14  
 apply\_visual\_color() (in module *yourdfpy.urdf*), 20  
 axis (*yourdfpy.urdf.Joint* attribute), 12

## B

base\_link (*yourdfpy.urdf.URDF* property), 15  
 Box (class in *yourdfpy.urdf*), 10  
 box (*yourdfpy.urdf.Geometry* attribute), 11

## C

Calibration (class in *yourdfpy.urdf*), 10  
 calibration (*yourdfpy.urdf.Joint* attribute), 12  
 center\_cfg (*yourdfpy.urdf.URDF* property), 15  
 cfg (*yourdfpy.urdf.URDF* property), 15  
 child (*yourdfpy.urdf.Joint* attribute), 12  
 clear\_errors() (*yourdfpy.urdf.URDF* method), 15  
 Collision (class in *yourdfpy.urdf*), 11  
 collision\_scene (*yourdfpy.urdf.URDF* property), 15  
 collisions (*yourdfpy.urdf.Link* attribute), 12  
 Color (class in *yourdfpy.urdf*), 11  
 color (*yourdfpy.urdf.Material* attribute), 12  
 contains() (*yourdfpy.urdf.URDF* method), 15  
 Cylinder (class in *yourdfpy.urdf*), 11  
 cylinder (*yourdfpy.urdf.Geometry* attribute), 11

## D

damping (*yourdfpy.urdf.Dynamics* attribute), 11  
 Dynamics (class in *yourdfpy.urdf*), 11  
 dynamics (*yourdfpy.urdf.Joint* attribute), 12

## E

effort (*yourdfpy.urdf.Limit* attribute), 12

errors (*yourdfpy.urdf.URDF* property), 15

## F

falling (*yourdfpy.urdf.Calibration* attribute), 11  
 filename (*yourdfpy.urdf.Mesh* attribute), 13  
 filename (*yourdfpy.urdf.Texture* attribute), 13  
 filename\_handler\_absolute2relative() (in module *yourdfpy.urdf*), 20  
 filename\_handler\_add\_prefix() (in module *yourdfpy.urdf*), 20  
 filename\_handler\_ignore\_directive() (in module *yourdfpy.urdf*), 20  
 filename\_handler\_ignore\_directive\_package() (in module *yourdfpy.urdf*), 21  
 filename\_handler\_magic() (in module *yourdfpy.urdf*), 21  
 filename\_handler\_meta() (in module *yourdfpy.urdf*), 21  
 filename\_handler\_null() (in module *yourdfpy.urdf*), 21  
 filename\_handler\_relative() (in module *yourdfpy.urdf*), 21  
 filename\_handler\_relative\_to\_urdf\_file() (in module *yourdfpy.urdf*), 22  
 filename\_handler\_relative\_to\_urdf\_file\_recursive() (in module *yourdfpy.urdf*), 22  
 friction (*yourdfpy.urdf.Dynamics* attribute), 11

## G

gazebo (*yourdfpy.urdf.Robot* attribute), 13  
 generate\_joint\_limit\_trajectory() (in module *yourdfpy.viz*), 22  
 Geometry (class in *yourdfpy.urdf*), 11  
 geometry (*yourdfpy.urdf.Collision* attribute), 11  
 geometry (*yourdfpy.urdf.Visual* attribute), 20  
 get\_transform() (*yourdfpy.urdf.URDF* method), 16

## H

hardware\_interfaces (*yourdfpy.urdf.Actuator* attribute), 10  
 hardware\_interfaces (*yourdfpy.urdf.TransmissionJoint* attribute), 14

**I**

`inertia` (`yourdfpy.urdf.Inertial` attribute), 11  
`Inertial` (class in `yourdfpy.urdf`), 11  
`inertial` (`yourdfpy.urdf.Link` attribute), 12

**J**

`Joint` (class in `yourdfpy.urdf`), 11  
`joint` (`yourdfpy.urdf.Mimic` attribute), 13  
`joint_map` (`yourdfpy.urdf.URDF` property), 16  
`joint_names` (`yourdfpy.urdf.URDF` property), 16  
`joints` (`yourdfpy.urdf.Robot` attribute), 13  
`joints` (`yourdfpy.urdf.Transmission` attribute), 14

**K**

`k_position` (`yourdfpy.urdf.SafetyController` attribute), 13  
`k_velocity` (`yourdfpy.urdf.SafetyController` attribute), 13

**L**

`length` (`yourdfpy.urdf.Cylinder` attribute), 11  
`Limit` (class in `yourdfpy.urdf`), 12  
`limit` (`yourdfpy.urdf.Joint` attribute), 12  
`Link` (class in `yourdfpy.urdf`), 12  
`link_map` (`yourdfpy.urdf.URDF` property), 16  
`links` (`yourdfpy.urdf.Robot` attribute), 13  
`load()` (`yourdfpy.urdf.URDF` static method), 16  
`lower` (`yourdfpy.urdf.Limit` attribute), 12

**M**

`main()` (in module `yourdfpy.viz`), 22  
`mass` (`yourdfpy.urdf.Inertial` attribute), 11  
`Material` (class in `yourdfpy.urdf`), 12  
`material` (`yourdfpy.urdf.Visual` attribute), 20  
`materials` (`yourdfpy.urdf.Robot` attribute), 13  
`mechanical_reduction` (`yourdfpy.urdf.Actuator` attribute), 10  
`Mesh` (class in `yourdfpy.urdf`), 13  
`mesh` (`yourdfpy.urdf.Geometry` attribute), 11  
`Mimic` (class in `yourdfpy.urdf`), 13  
`mimic` (`yourdfpy.urdf.Joint` attribute), 12  
`module`  
    `yourdfpy`, 23  
    `yourdfpy.urdf`, 10  
    `yourdfpy.viz`, 22  
`multiplier` (`yourdfpy.urdf.Mimic` attribute), 13

**N**

`name` (`yourdfpy.urdf.Actuator` attribute), 10  
`name` (`yourdfpy.urdf.Collision` attribute), 11  
`name` (`yourdfpy.urdf.Joint` attribute), 12  
`name` (`yourdfpy.urdf.Link` attribute), 12  
`name` (`yourdfpy.urdf.Material` attribute), 13

`name` (`yourdfpy.urdf.Robot` attribute), 13  
`name` (`yourdfpy.urdf.Transmission` attribute), 14  
`name` (`yourdfpy.urdf.TransmissionJoint` attribute), 14  
`name` (`yourdfpy.urdf.Visual` attribute), 20  
`num_actuated_joints` (`yourdfpy.urdf.URDF` property), 17  
`num_dofs` (`yourdfpy.urdf.URDF` property), 17

**O**

`offset` (`yourdfpy.urdf.Mimic` attribute), 13  
`origin` (`yourdfpy.urdf.Collision` attribute), 11  
`origin` (`yourdfpy.urdf.Inertial` attribute), 11  
`origin` (`yourdfpy.urdf.Joint` attribute), 12  
`origin` (`yourdfpy.urdf.Visual` attribute), 20

**P**

`parent` (`yourdfpy.urdf.Joint` attribute), 12  
`parse_args()` (in module `yourdfpy.viz`), 22

**R**

`radius` (`yourdfpy.urdf.Cylinder` attribute), 11  
`radius` (`yourdfpy.urdf.Sphere` attribute), 13  
`rgba` (`yourdfpy.urdf.Color` attribute), 11  
`rising` (`yourdfpy.urdf.Calibration` attribute), 11  
`Robot` (class in `yourdfpy.urdf`), 13  
`run()` (in module `yourdfpy.viz`), 22

**S**

`safety_controller` (`yourdfpy.urdf.Joint` attribute), 12  
`SafetyController` (class in `yourdfpy.urdf`), 13  
`scale` (`yourdfpy.urdf.Mesh` attribute), 13  
`scene` (`yourdfpy.urdf.URDF` property), 17  
`setup_logging()` (in module `yourdfpy.viz`), 23  
`show()` (`yourdfpy.urdf.URDF` method), 18  
`size` (`yourdfpy.urdf.Box` attribute), 10  
`soft_lower_limit` (`yourdfpy.urdf.SafetyController` attribute), 13  
`soft_upper_limit` (`yourdfpy.urdf.SafetyController` attribute), 13  
`Sphere` (class in `yourdfpy.urdf`), 13  
`sphere` (`yourdfpy.urdf.Geometry` attribute), 11  
`split_along_joints()` (`yourdfpy.urdf.URDF` method), 18

**T**

`Texture` (class in `yourdfpy.urdf`), 13  
`texture` (`yourdfpy.urdf.Material` attribute), 13  
`Transmission` (class in `yourdfpy.urdf`), 14  
`transmission` (`yourdfpy.urdf.Robot` attribute), 13  
`TransmissionJoint` (class in `yourdfpy.urdf`), 14  
`type` (`yourdfpy.urdf.Joint` attribute), 12  
`type` (`yourdfpy.urdf.Transmission` attribute), 14

## U

`update_cfg()` (*yourdfpy.urdf.URDF* method), 18  
`upper` (*yourdfpy.urdf.Limit* attribute), 12  
`URDF` (class in *yourdfpy.urdf*), 14  
`URDFAttributeValueError`, 19  
`URDFBrokenRefError`, 19  
`URDFError`, 19  
`URDFIncompleteError`, 19  
`URDFMalformedError`, 19  
`URDFSaveValidationError`, 19  
`URDFUnsupportedError`, 19

## V

`validate()` (*yourdfpy.urdf.URDF* method), 18  
`validate_filenames()` (*yourdfpy.urdf.URDF* method), 18  
`validation_handler_strict()` (in module *yourdfpy.urdf*), 22  
`velocity` (*yourdfpy.urdf.Limit* attribute), 12  
`viewer_callback()` (in module *yourdfpy.viz*), 23  
`Visual` (class in *yourdfpy.urdf*), 19  
`visuals` (*yourdfpy.urdf.Link* attribute), 12

## W

`write_xml()` (*yourdfpy.urdf.URDF* method), 18  
`write_xml_file()` (*yourdfpy.urdf.URDF* method), 19  
`write_xml_string()` (*yourdfpy.urdf.URDF* method), 19

## Y

`yourdfpy`  
    module, 23  
`yourdfpy.urdf`  
    module, 10  
`yourdfpy.viz`  
    module, 22

## Z

`zero_cfg` (*yourdfpy.urdf.URDF* property), 19