Appendix A

URDF to MBS Converter

As discussed in section 3.1, the gait controllers cannot directly be developed on the real robots. A software model describing the kinematics and dynamics of the robots is then required. The Universal Robotic Description Format (URDF) is an XML file format used in ROS which describes robots. However, we would like this model to be supported by Robotran (the simulation environment used throughout this thesis) which is not the case for the URDF format. Therefore, the software model should be represented by another XML format called MBS (MultiBody System). This MBS description can be either manually edited or generated thanks to MBSysPad, a graphical user interface. It is, in any case, a long and tedious work. The development of a tool converting an available description format, containing all the desired data, into an MBS file would be a considerable saving in time.
We will first present the URDF specification. Then it will be compared to the MBS format. After that, the conversion logic will be explained with an emphasis given to the adaptations needed. We will finally validate the converter by testing it with multiple examples.

A.1 Unified Robot Description Format (URDF)

The Unified Robot Description Format (URDF) is an XML format used in ROS[1] that represent a robot model. This specification is quite general, but it cannot describe all robots. The main limitation is that only tree structures can be represented, ruling out all parallel robots. The specification also assumes that the robot only consists of rigid links connected by joints. Flexible elements are thus not supported. Humanoid robots are a good example of robots that can easily be described with the URDF format. The URDF description of a lot of humanoid robots (NAO, Atlas, CoMan and WalkMan included) can be found on the internet.

The specification covers:

- Kinematics and dynamics of the robot
- Collision model of the robot
- Visual representation of the robot

The URDF description of a robot consists of a set of joint elements connecting a set of links together. Because URDF can only represent tree structures, a typical URDF robot description looks like illustrated in Figure A.2.

[1]Robot Operating System (ROS) is a collection of software frameworks for robot software development. For more information, see `http://www.ros.org`

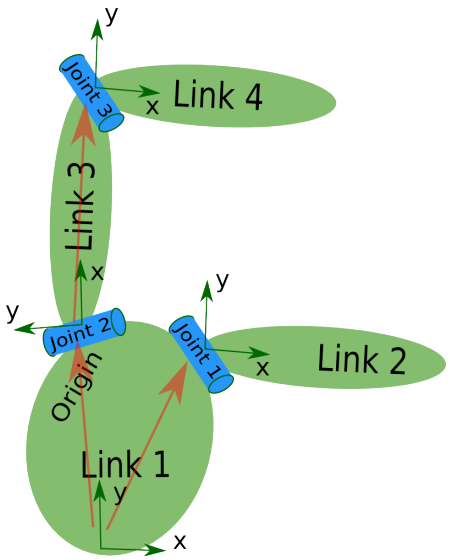
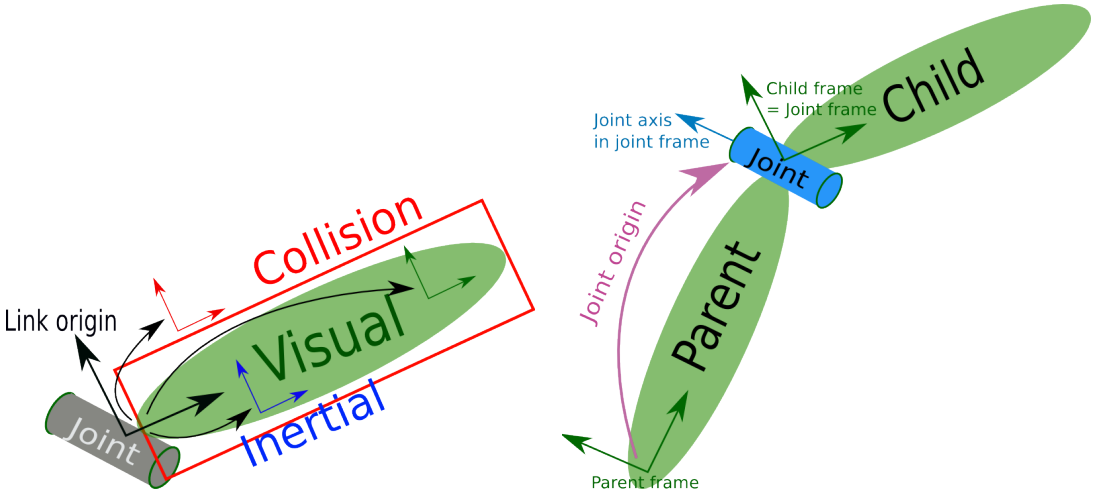


Figure A.1: Basic schematic example of a tree structure URDF specification (Figure taken from `http://wiki.ros.org/urdf`)

A.1.1 Link elements

The inertial, visual and collision properties of each rigid body of the multibody system are described by a link element. As shown in Figure A.2a, inertial, visual and collision properties are defined in 3 different frames. As presented in Listing A.1, three subelements ("inertial", "visual" and "collision") describe the corresponding type of properties.



(a) Schematic example of a link element with its different frames

(b) Schematic example of a joint element with its different frames

Figure A.2: Figures taken from `http://wiki.ros.org/urdf`

Inertial element

The inertial element has three subelements :

- origin : this is the pose of the inertial reference frame, relative to the link reference frame. This subelement has two attributes :
  - xyz : the center of gravity described in the link reference frame

  - rpy : the roll, pitch and yaw angles (in radians) describing the transformation from the link reference frame to the inertial reference frame. This transformation is a "fixed frame" transformation

- mass : the mass of the body
- inertia : the rotational inertia matrix, represented in the inertial frame

Visual element

Two subelements of the visual element are important in this case :

- origin : this is the pose of the visual element with respect to the reference frame of the link. This subelement has two attributes :
  - xyz : the position of the visual element along the x, y and z axis in the reference frame of the link
  - rpy : the roll, pitch and yaw angles (in radians) describing the orientation of the visual element relative to the link reference frame. This transformation is a "fixed frame" transformation
- geometry : the visual appearance of the element. This can be a geometric figure or a mesh file (a Collada .dae file or a even .stl file).

Collision element

This element has the same structure as the visual one and is not of great interest for the rest of this chapter.

Listing A.1: Example of a URDF link element

```
<link name="name">
   <inertial>
     <origin xyz="x y z" rpy="r p y"/>
     <mass value="m"/>
     <inertia ixx="ixx"  ixy="ixy"  ixz="ixz" iyy="iyy" iyz="iyz" izz="izz" />
   </inertial>

   <visual>
     <origin xyz="x y z" rpy="r p y" />
     <geometry>
        <mesh filename="path" scale = "x y z"/>
     </geometry>
   </visual>

   <collision>
     <origin xyz="x y z" rpy="r p y"/>
     <geometry>
       <cylinder radius="r" length="l"/>
     </geometry>
   </collision>
 </link>
```

A.1.2 Joint elements

The kinematics of each joint is described by a joint element whose type can be :

- revolute/continuous : joint that rotates along the axis
- prismatic : sliding joint that slides along the axis

- fixed : not really a joint because it cannot move. All degrees of freedom are locked
- floating : joint allowing motion for all 6 degrees of freedom
- planar : joint allowing motion in a plane perpendicular to the axis

The first three types are the most common. As presented in Listing A.2, a joint element has four important subelements.

- origin : 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 Figure A.2b. It has two attributes :
  - xyz : the anchor point of the joint on the parent link
  - rpy : the roll, pitch and yaw angles (in radians) describing the transformation from the parent link to the child link. This transformation is a "fixed frame" transformation
- parent : the parent link
- child : the child link
- axis : the joint axis specified in the child frame

Listing A.2: Example of a URDF joint element

```
<joint name="name" type="type">
    <origin xyz="x y z" rpy="r p y"/>
    <parent link="parent"/>
    <child link="child"/>
    <axis xyz="x y z"/>
 </joint>
```

A.2 Robotran MultiBody System (MBS)

As written above, the MBS (MultiBody System) format is the one supported by Robotran, the multibody simulation environment used throughout this thesis. Like Robotran, this description format was developed at the Université catholique de Louvain (UCL) within the Center for Research in Mechatronics (CEREM).
MBS files can be generated either manually by editing the XML type file or thanks to a graphical user interface : MBSysPad. MBSysPad allows the user to draw a diagram of segments linked by joints as shown in Figure A.3.
A MBS specification describes a mechanical system composed of rigid bodies that are connected together by joints which can be subject to rotational and/or translational displacements.

The MBS format covers :

- Kinematic and dynamic description of the robot
- Visual representation of the robot

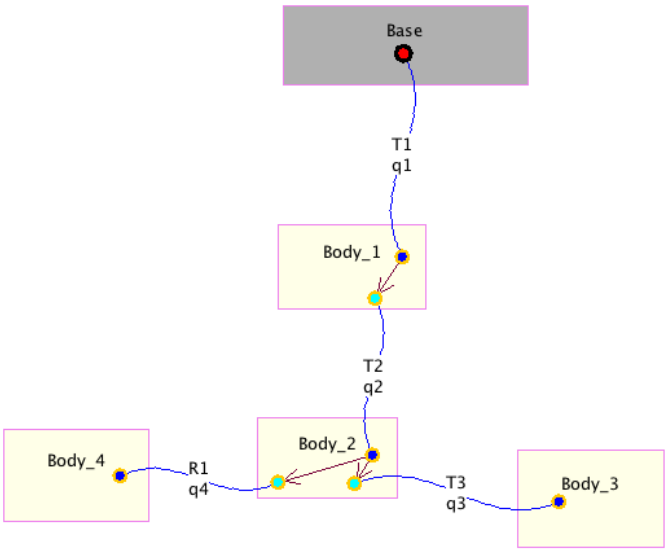


Figure A.3: Example of MBS made with MBSysPad

The MBS description, like the URDF description, consists of a set of bodies connected together by a set of joints in a tree-like structure. Each body has a unique parent body but can have several child bodies. Unlike URDF structure, MBS structure allows a joint to be attached to another joint.
For each body, there is a body-fixed frame. This frame moves with the body and is fixed at the position of the parent joint.
The MBS bodytree is only based on one type of element : the bodies.
A body element generally has eight important subelements :

- bodyname : the name of the body
- parent : the parent of the body. It has two subelements :
  - bodyname : the name of the parent body
  - pointname : the point, on the parent body, where the joint binding both bodies is attached (the anchor point)
- joint : the joint(s) binding the body to its parent. There are four subelements for each joint :
  - jointname : the name of the joint
  - type : the type of joint, revolute (R1, R2, R3) or prismatic (T1, T2, T3)
  - nature : the nature of the joint :
    - dependent
    - independent
    - forced driven
  - graphics : this field has no physical meaning, it defines the way the joint is drawn in the MBSysPad graphical interface
- mass : the mass of the body
- com : the center of mass of the body relative to the body-fixed frame

- inertia : the rotational inertia matrix, represented in the body-fixed frame
- point : the anchor point(s) of the body relative to the body-fixed frame
- graphics : this field has no physical meaning. It has two subelements :
  - x2D : the way the body is drawn in the MBSysPad graphical interface
  - x3D : the visual properties of the body. Some precisions about three fields (the others are trivial)
    * position : the pose of the visual element in the body-fixed frame
    * rotation : the roll, pitch and yaw angles (in radians) describing the orientation of the visual element relative to the body-fixed frame. This transformation is a "current frame" transformation.
    * url : the path leading to the mesh file (a .wrl file).

The root of this bodytree is always a base element whose structure consists of three elements :

- gravity : the x, y and z components of the gravity
- baseBodyName : the name of the base
- graphics : this field has no physical meaning. It has 2 subelements :
  - x2D : the way the base is drawn in the MBSysPad graphical interface
  - x3D : the visual properties of the base

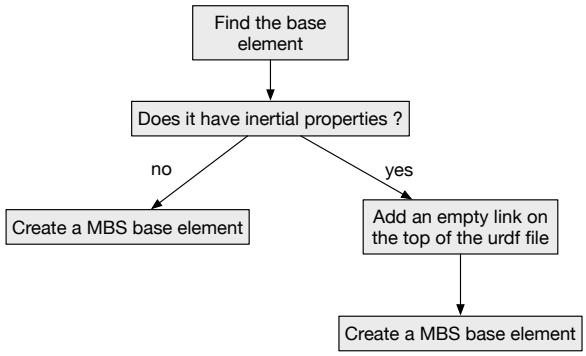
A.3 Python tool

As presented in the two previous sections, URDF and MBS are two different ways of specifying the kinematic and dynamic description of robots (both of them also cover the visual representation of the robot). All the information needed to create a MBS description can be found in the corresponding URDF description but sometimes needs to be adapted due to conventions differences.
We developed a converting tool in Python aiming to create a MBS file, to retrieve the needed data in the URDF file and to translate these data following the MBS conventions.
This converter is as general as possible but it was particularly developed for robots having a tree-like structure, typically for humanoid robots.

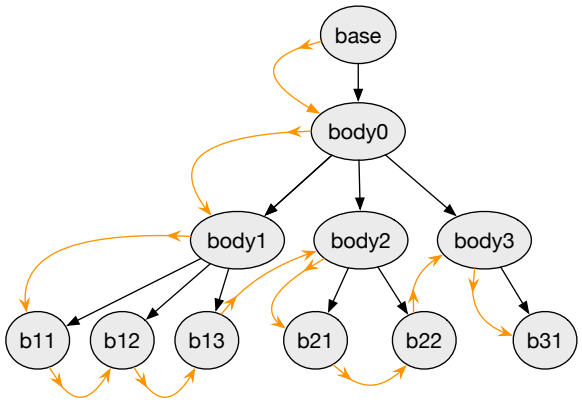
A.3.1 Converter development

Creation of the base element

As written above, the root of a MBS bodytree is always a base element. A proper MBS base element does not have any parent and does not have any inertial property. In the URDF format, the root link sometimes has inertial properties. The converter must, in that case, modify the initial URDF file by adding a URDF link without inertial properties on top of the tree structure. The base creation process is summed up in Figure A.4a.
As stated in section 3.1, the humanoid robots models are based on a floating base representation. The base body is then free to move and this is represented by a 6-DOF joint (three prismatic and three revolute joints about the XYZ axes) between the robot and the world inertial frame. These joints will appear when creating the first body element of the bodytree (the one attached to the base element).



(a) Base creation process



(b) Illustration of the descending order in which the URDF tree structure is browsed

Creation of the bodytree

To create the bodytree, the URDF tree structure is browsed in descending order, in the same way as in Figure A.4b. For each URDF link found (except the null mass links which are neglected), a MBS body is added in the bodytree following the steps below :

1. Retrieve the inertial properties of the URDF link and adapt them to the MBS body conventions

In both descriptions, the mass is expressed in kilograms and the center of mass is described in the body frame. There is then no need to adapt these data.
However, as stated in section A.1.1, the URDF rotational inertia matrix is represented in the inertial frame while the MBS rotational inertia matrix is represented in the body reference frame. The inertial frame is the frame obtained by applying a "roll-pitch-yaw" transformation in "fixed frame" to the body reference frame[2]. Therefore, the components of the MBS rotational inertia matrix need to be adapted.
Considering $\{\hat{I}\}$ and $\{\hat{B}\}$, the inertial and body frames respectively,

$$[\hat{B}] = R[\hat{I}]$$

$$[\hat{I}] = R^T[\hat{B}]$$

R being the transformation matrix for the roll-pitch-yaw transformation in fixed frame.

$$R = R(Z, y)R(Y, p)R(X, r)$$

$$R = \begin{pmatrix} cos(y)cos(p) & -sin(y)cos(r) + cos(y)sin(p)sin(r) & sin(y)sin(r) + cos(y)sin(p)cos(r) \\ sin(y)cos(p) & cos(y)cos(r) + sin(y)sin(p)sin(r) & -cos(y)sin(r) + sin(y)sin(p)cos(r) \\ -sin(p) & cos(p)sin(r) & cos(p)cos(r) \end{pmatrix}$$

Considering $T_{\{\hat{I}\}}$, the rotational inertia matrix described in frame $\{\hat{I}\}$,

$$T_{\{\hat{I}\}} = \begin{pmatrix} i_{xx} & i_{xy} & i_{xz} \\ i_{xy} & i_{yy} & i_{yz} \\ i_{xz} & i_{yz} & i_{zz} \end{pmatrix}$$

then, $T_{\{\hat{B}\}}$, the corresponding rotational inertia matrix described in frame $\{\hat{B}\}$ can be computed as follows :

$$T_{\{\hat{B}\}} = RT_{\{\hat{I}\}}R^T$$

[2]The roll, pitch and yaw angles (in radians) can be found in the URDF link element, under *inertial>origin>rpy*

2. Retrieve information on the joint binding the URDF link to its parent

The next thing the converter does is to check if the transformation between the URDF parent frame and the URDF joint frame has rotation components[3]. If it is the case, then additional "orientation" forced driven joints are needed before the real DOF because the axis of rotation is described in the joint frame (which is equivalent to the child frame as shown in Figure A.2b) and not in the parent frame.

The URDF description gives the angles r (for roll), p (for pitch) and y (for yaw) for the transformation in fixed frame (rotation of r around X, p around Y and y around Z) . However, successive joints in the MBS description correspond to transformations in current frame. Therefore, we need to find the equivalent angles α, β and γ (rotation of α around X, β around Y and γ around Z) for a transformation in current frame.

In fixed frame[4], we have :

$$R = R(Z, y)R(Y, p)R(X, r)$$

$$R = \begin{pmatrix} cos(y)cos(p) & -sin(y)cos(r) + cos(y)sin(p)sin(r) & sin(y)sin(r) + cos(y)sin(p)cos(r) \\ sin(y)cos(p) & cos(y)cos(r) + sin(y)sin(p)sin(r) & -cos(y)sin(r) + sin(y)sin(p)cos(r) \\ -sin(p) & cos(p)sin(r) & cos(p)cos(r) \end{pmatrix}$$

In current frame[5], we have :

$$R = R(X, \alpha)R(Y, \beta)R(Z, \gamma)$$

$$R = \begin{pmatrix} cos(\beta)cos(\gamma) & -sin(\gamma)cos(\beta) & sin(\beta) \\ sin(\alpha)sin(\beta)cos(\gamma) + cos(\alpha)sin(\gamma) & -sin(\alpha)sin(\beta)sin(\gamma) + cos(\alpha)cos(\gamma) & -sin(\alpha)cos(\beta) \\ -sin(\beta)cos(\alpha)cos(\gamma) + sin(\alpha)sin(\gamma) & sin(\beta)sin(\gamma)cos(\alpha) + sin(\alpha)cos(\gamma) & c(\alpha)c(\beta) \end{pmatrix}$$

Using these two matrices it can be stated that :

$$\frac{-r_{23}}{r_{33}} = tan(\alpha) \rightarrow \alpha = atan2(cos(y)sin(r) - sin(y)sin(p)cos(r), cos(p)cos(r))$$

$$\frac{-r_{12}}{r_{11}} = tan(\gamma) \rightarrow \gamma = atan2(cos(r)sin(y) - sin(r)sin(p)cos(y), cos(p)cos(y))$$

$$\frac{r_{13}}{\frac{1}{cos(\alpha)}r_{33}} = tan(\beta) \rightarrow \beta = atan2(sin(r)sin(y) + cos(y)sin(p)cos(r), \frac{1}{cos(\alpha)}cos(p)cos(r))$$

Now, the converter can create the real DOF joint by retrieving the type of the joint and its axis. The translation between the URDF and the MBS joints is trivial and is summed up in Table A.1.

Table A.1

URDF (type and axis)	MBS (type and nature)
Revolute/continuous of axis (1, 0, 0)	R1 independent
Revolute/continuous of axis (0, 1, 0)	R2 independent
Revolute/continuous of axis (0, 0, 1)	R3 independent
Prismatic of axis (1, 0, 0)	T1 independent
Prismatic of axis (0, 1, 0)	T2 independent
Prismatic of axis (0, 0, 1)	T3 independent
Floating base	T1T2T3R1R2R3
Fixed	T3 forced driven to 0[m]
Planar	not considered

[3]If, under the URDF joint element, the field *origin>rpy* has non-zero components

[4]In fixed frame, rotation matrices are pre-multiplied

[5]In current frame, rotation matrices are post-multiplied

However, while MBS joints can only move along/around axis XYZ, URDF joints axis can take a large number of values. Therefore, the converter needs one, three or five MBS joints (depending on the case) to convert one URDF joint. The joints chain respects the following structure :

1. One or two "forced driven" joints to allow the real DOF joint to rotate around the desired axis.
2. One independent joint, the real DOF joint.
3. One or two "forced driven" joints to get back to the joint frame.

The different cases are presented in Table A.2.

Table A.2

URDF axis	MBS type	MBS nature	MBS initial value
(-1, 0, 0)	R3 or R2	forced driven	π
	+ R1	independent	
	+ R3 or R2	forced driven	-π
(0, -1, 0)	R1 or R3	forced driven	π
	+ R2	independent	
	+ R1 or R3	forced driven	-π
(0, 0, -1)	R1 or R2	forced driven	π
	+ R3	independent	
	+ R1 or R2	forced driven	-π
(0, y, z)	R1	forced driven	$atan2(z, y)$
	+ R2	independent	
	+ R1	forced driven	-$atan2(z, y)$
(x, 0, z)	R2	forced driven	$atan2(x, z)$
	+ R3	independent	
	+ R2	forced driven	-$atan2(x, z)$
(x, y, 0)	R3	forced driven	$atan2(y, x)$
	+ R1	independent	
	+ R3	forced driven	-$atan2(y, x)$
(x, y, z)	R3	forced driven	$atan2(y, x)$
	+ R2	forced driven	-$atan2(z, x)$
	+ R1	independent	
	+ R2	forced driven	$atan2(z, x)$
	+ R3	forced driven	-$atan2(y, x)$

3. Retrieve the coordinates of the anchor points

The anchor points of the MBS body are the anchor points specified in the URDF joint elements whose parent link is the corresponding URDF link.

4. Retrieve the visual properties of the URDF link and adapt them to the MBS body conventions

In both description, the mesh file, the position, the scaling and the orientation of the visual element are given.

- The mesh file simply needs to be converted (by the user) from .dae/.stl to .wrl.
- For both description, the position and the scaling are described in the body-fixed frame.

- The orientation is described :
  - by three fixed frame rotations (angle r around X, angle p around Y and angle y around Z) for the URDF description.
  - by three current frame rotations (rotation of α around X, β around Y and γ around Z) for the MBS description.

  To find the angles α, β and γ equivalent to the angles r, p and y, the reasoning is exactly the same as in the second step. This gives :

$$\alpha = atan2(cos(y)sin(r) - sin(y)sin(p)cos(r), cos(p)cos(r))$$

$$\gamma = atan2(cos(r)sin(y) - sin(r)sin(p)cos(y), cos(p)cos(y))$$

$$\beta = atan2(sin(r)sin(y) + cos(y)sin(p)cos(r), \frac{1}{cos(\alpha)}cos(p)cos(r))$$

5. Give 2D graphical properties to each MBS element

Each body is represented by a rectangle box in MBSysPad. To obtain an organized and easy to handle diagram in MBSysPad, the position of each rectangle box is assigned in such a way that it never overlaps an other box. To do so, each body is given (x,y) coordinates in an imaginary two-dimensional array. The base body is located in the upper-left cell ((0,0)).
Considering Figure A.5, the (x,y) coordinates evolution is as follows :

- orange arrows (parent to child) : (x,y) → (x, y+1)
- purple arrows (brother to brother) : (x,y) → (x+1, y)
- green arrows (nephew to uncle) : (x,y) → (x+1, y-1)

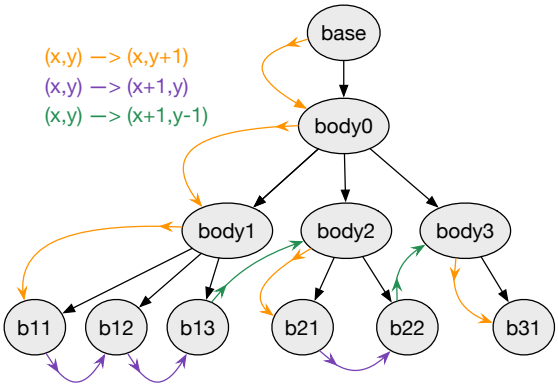


Figure A.5: Attribution rule for the bodies graphical positions in MBSysPad

This gives the result of Figure A.6.

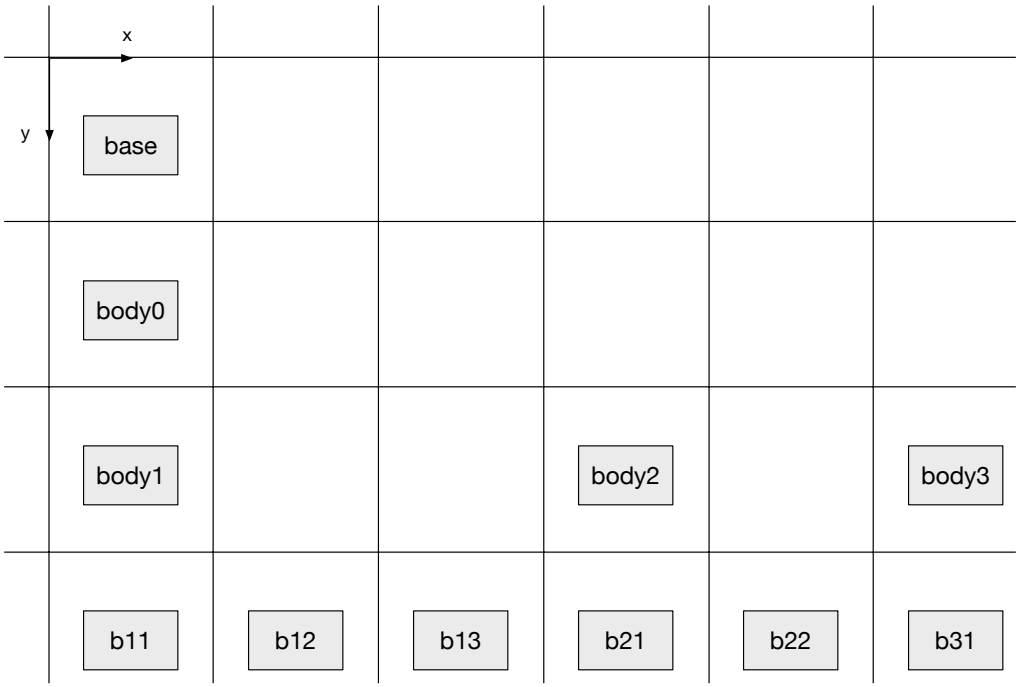


Figure A.6: Typical graphical layout created by the converter for the tree-like structure example of Figure A.5

A.3.2 Converter validation

In order to test and validate the developed converter, we checked the outputs produced by the Python program for four different inputs (the URDF files of robots Atlas, NAO, WalkMan and iCub).
In Figures A.7, A.9, A.12 and A.14, we see that, for all the robots tested, the graphical interface is organized as presented in Figure A.6. This is clear for the user who can rearrange it afterwards (e.g., by drawing a human-like diagram for the sake of clarity, see section 3.1).
In Figures A.8, A.10, A.13 and A.15, we see that the 3D Model of MBSysPad looks coherent with the expected aspects of the different robots.

Results for Atlas :

The URDF file describing the Atlas robot was found on the internet : `https://bitbucket.org/osrf/drcsim/src/583228ebd72677c36a393b3260f0c5f0bc7b0f87?at=default`

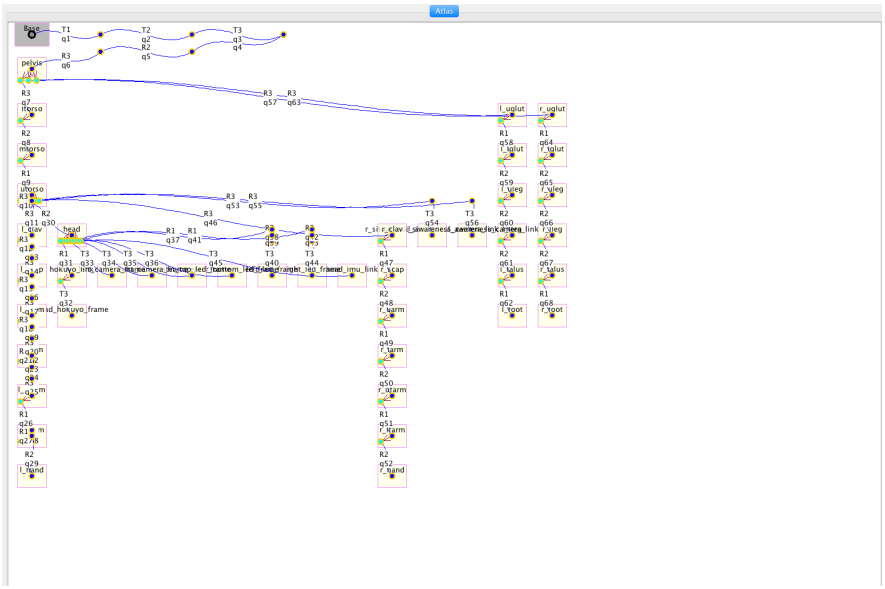


Figure A.7: Screenshot of the MBSysPad graphical interface for the generated Atlas MBS file

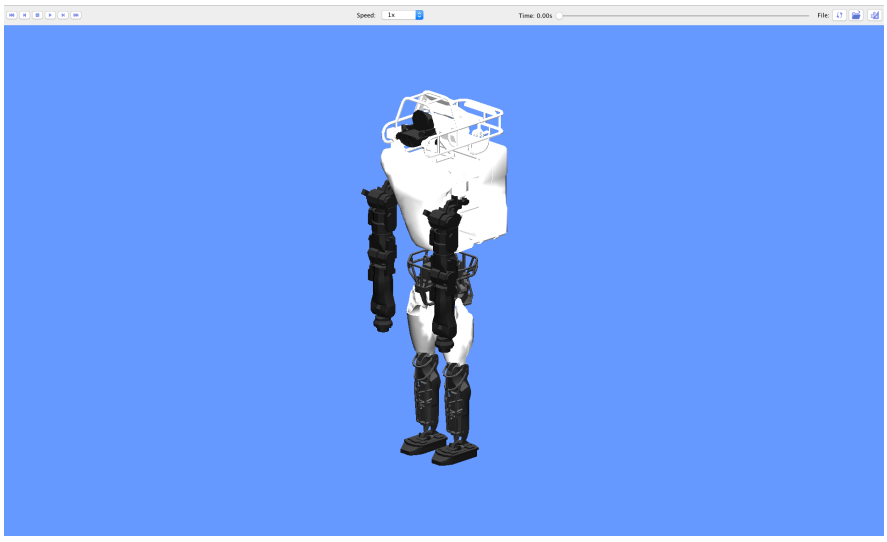


Figure A.8: Screenshot of the MBSysPad Animation 3D Model for the generated Atlas MBS file

Results for NAO :

The URDF file describing the NAO robot was found on the internet : `https://github.com/ros-naoqi/nao_robot/blob/master/nao_description/urdf/naoV50_generated_urdf/nao.urdf`

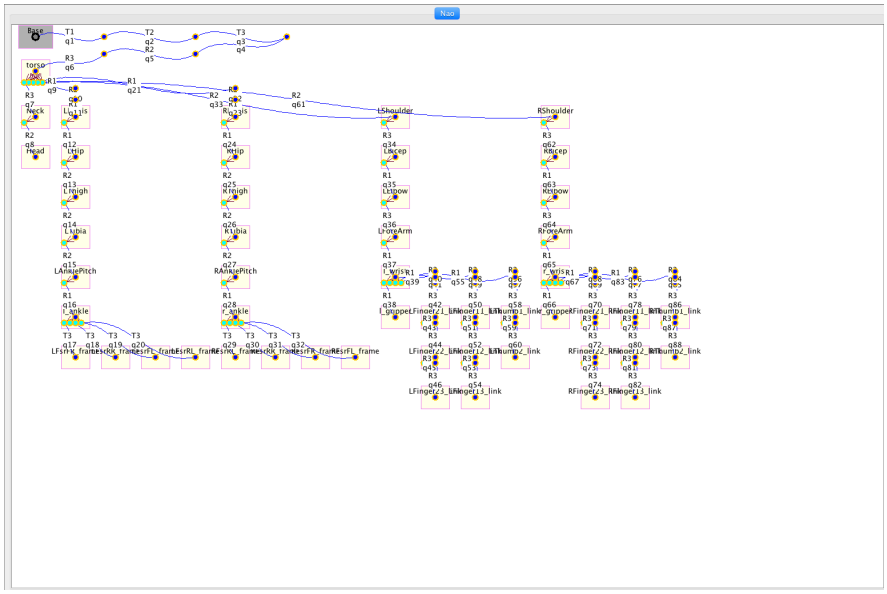


Figure A.9: Screenshot of the MBSysPad graphical interface for the generated NAO MBS file

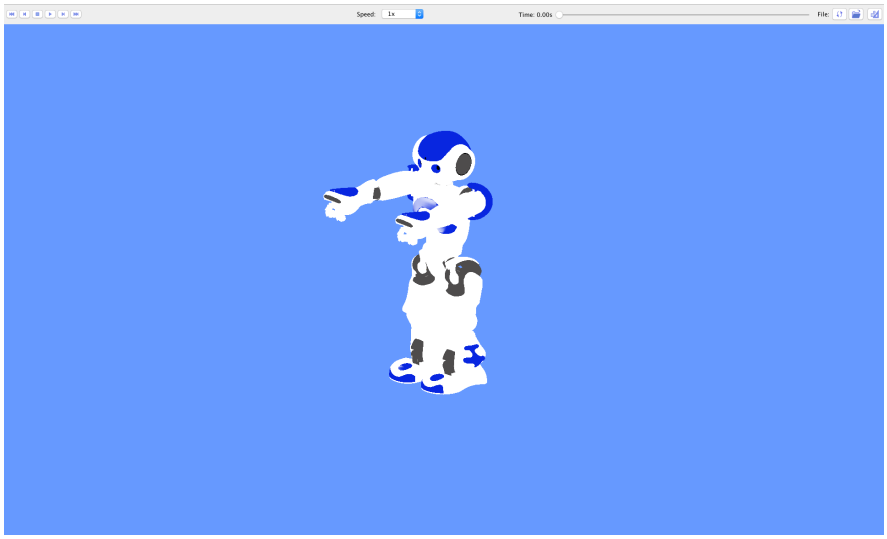
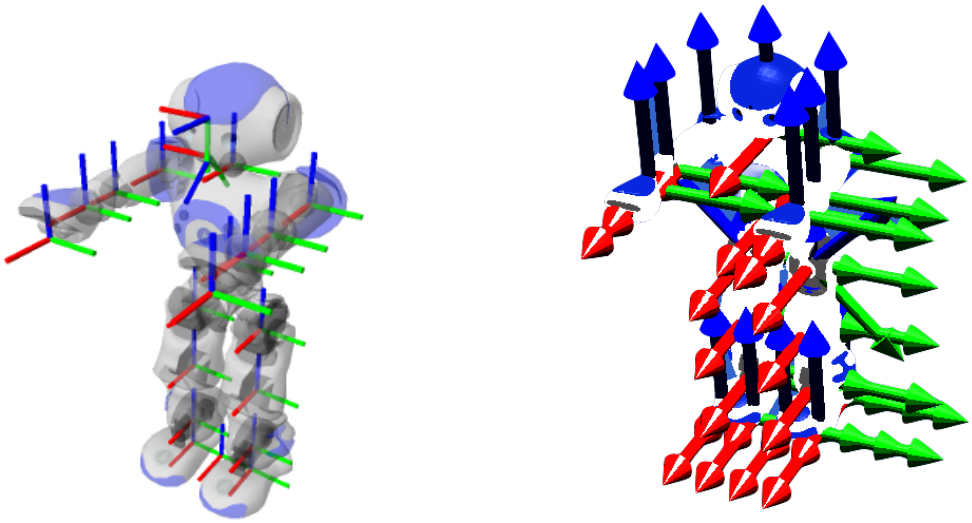


Figure A.10: Screenshot of the MBSysPad Animation 3D Model for the generated NAO MBS file

As an additional check, Figure A.11 compares the expected NAO model (provided by ROS) and the output of the converter. The joint frames definitions are exactly the same.



(a) Expected NAO model provided by ROS (figure taken from `http://wiki.ros.org/nao_description`)

(b) Screenshot of the MBSysPad Animation 3D Model for the generated NAO MBS file with apparent joint frames

Figure A.11: Comparison between the expected NAO model and the output of the converter

Results for WalkMan :

The URDF file describing the WalkMan robot was found on the internet : `https://gitlab.robotran.be/walkman/walkman_robotran/blob/master/dataR/urdf%20files/bigman.urdf`

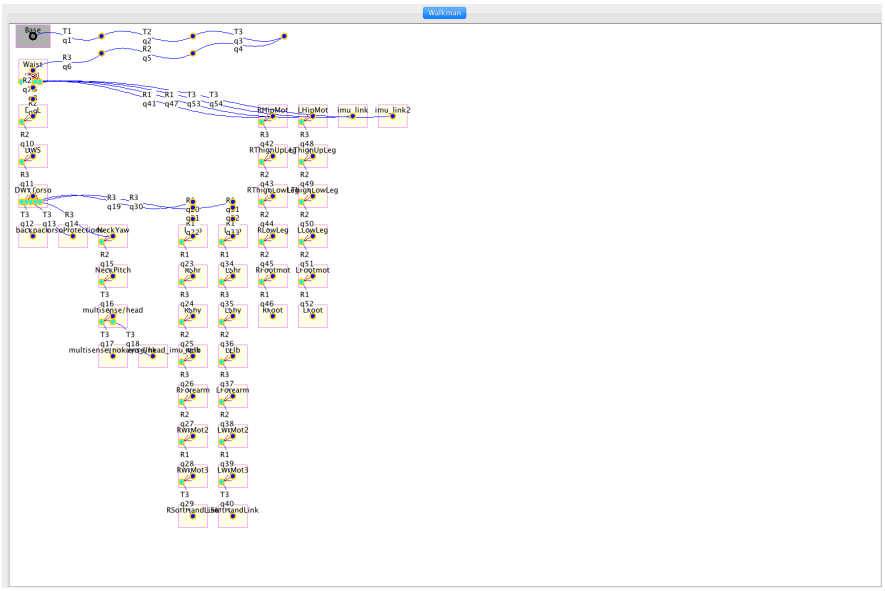


Figure A.12: Screenshot of the MBSysPad graphical interface for the generated WalkMan MBS file

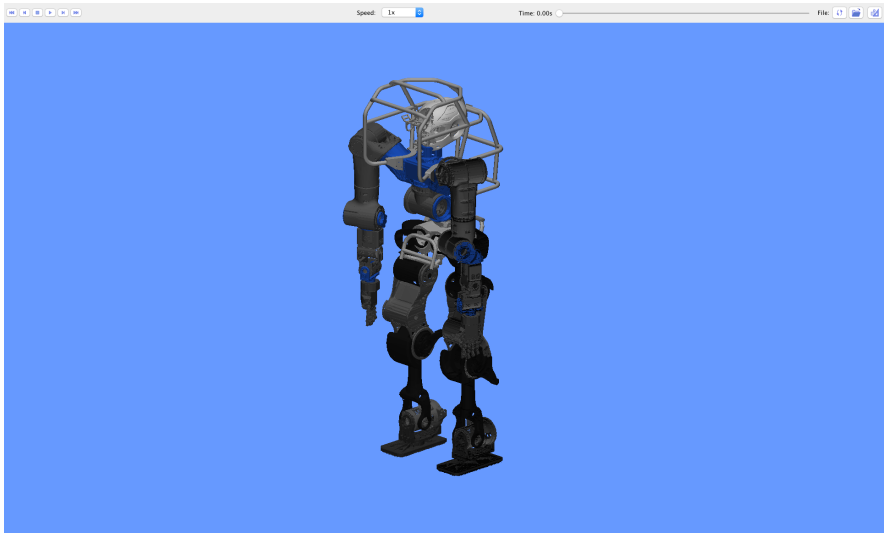


Figure A.13: Screenshot of the MBSysPad Animation 3D Model for the generated WalkMan MBS file

Results for iCub :

The URDF file describing the iCub robot was found on the internet : `https://github.com/robotology-playground/icub-model-generator/blob/master/generated/urdf/iCubParis02/icub.urdf`

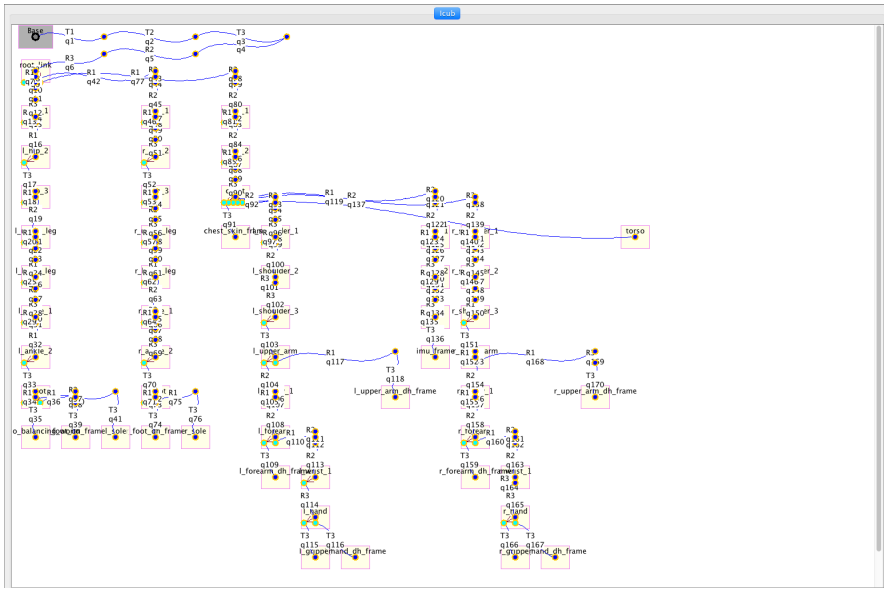


Figure A.14: Screenshot of the MBSysPad graphical interface for the generated iCub MBS file

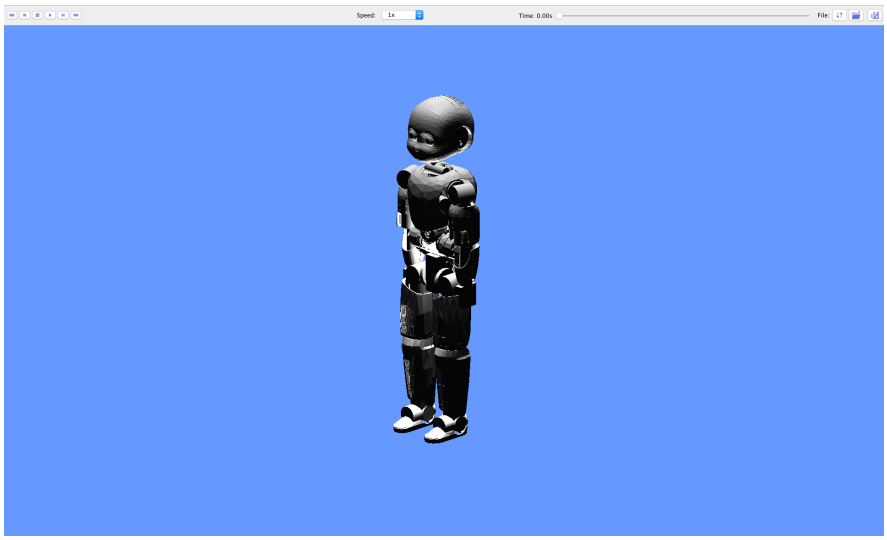


Figure A.15: Screenshot of the MBSysPad Animation 3D Model for the generated iCub MBS file