Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 200 (2022) 858–864

Procedia Computer Science

www.elsevier.com/locate/procedia

3rd International Conference on Industry 4.0 and Smart Manufacturing

An approach for automatic generation of the URDF file of modular robots from modules designed using SolidWorks

Maddalena Feder[a,b,*], Andrea Giusti[a], Renato Vidoni[b]

[a] *Fraunhofer Italia Research, Via A. Volta 13, Bolzano 39100, Italy*
[b] *Free University of Bozen-Bolzano, Piazza Università 1, Bolzano 39100, Italy*

Abstract

Modular robot manipulators promise to enhance current automation practice by providing higher flexibility and quick recovery in case of failures with respect to fixed-structure robots. The configuration of the modular robot control for each new assembly, to account for the new system kinematics and dynamics, can be time consuming and requires robot modelling expertise. We propose an approach for automatically generating the Unified Robot Description Format (URDF) file of modular robot manipulators, starting from the kinematic and dynamic descriptions expressed following the URDF of the single modules they can be composed of. The approach has been implemented and numerically verified by exploiting off-the-shelf software tools from Robot Operating System (ROS) libraries.

Keywords: Modular reconfigurable robots; Industry 4.0; Robot Operating System; URDF; Kinematics

1. Introduction

Current competitive scenarios in industry push companies to implement modern technologies for securing their position among competitors. In this competitive scenario of the fourth industrial revolution, named Industry 4.0, new trends emerged [1], involving e.g., advanced robotics and big data analytics, which are transforming the whole manufacturing sector. The introduction of the Cyber-Physical Systems (CPS) has supported the integration of capabilities from humans and machines. The same is occurring also in other sectors, e.g., agriculture [2]. Advanced robotics is a central aspect of Industry 4.0 for seeking higher flexibility in production. Among other solutions, modular reconfigurable robots are particularly promising since they allow different and quick rearrangements of the robotic elements [3]. Consequently, the flexibility and the possibility to swiftly adapt the robot to different tasks, working spaces and constraints increases [4]. Another advantage consists in quickly repairing damaged robots by only substituting their

* Corresponding author.
E-mail address: maddalena.feder@gmail.com

1877-0509
10.1016/j.procs.2022.01.283

broken modules [5]. The automatic generation of an assembled modular robot description from its modules is important both for control [6] and for optimizing the selection of the assembly for given tasks [5]. With respect to existing modelling approaches, which model modular robot manipulators considering extensions of the standard and modified Denavit-Hartenberg (DH) conventions [6, 7] or the Product of Exponential (PoE) formulation [8], we target simplicity of implementation for obtaining an assembled modular robot description as an Unified Robot Description Format (URDF) file, that can be readily used for applications with software tools from Robot Operating System (ROS) [9] libraries.

For practical realization of a description for modelling kinematics and dynamics of robots within ROS, one can enjoy the availability of the plugin for the Computer Aided Design (CAD) tool: SolidWorks to URDF exporter [1]. As described in [10] and in [11], the design of a single SolidWorks assembly results, thanks to the plugin, in a single assembled robot. Therefore, the change of a module implies the redesign of the robot and a new exportation to URDF. Our work aims at removing the need for a robot redesign and at supporting practitioners by developing a solution for an automatic generation of the URDF file of modular robots. Then, the developed approach exploits the mentioned plugin to create independent robot modules, whose URDF files can then be combined together in different orders to result in several robot manipulators.

The remainder of this paper is structured as follows. Essentials of modelling robot modules and the URDF are described in Section 2 and the main results with the proposed conventions are presented in Section 3. Finally, discussions and conclusions are presented in Section 4 and Section 5, respectively.

2. Material and Methods

2.1. Essentials of modelling robot modules

The basic element of a modular robot is represented by the module itself. As detailed e.g., in [3], each module can be used as a building block for composing a serial robot arm by means of standardized connectors. When self programming is desired, each module should store all its kinematic, dynamic and geometry properties in itself. Thanks to this practice, after assembling the modules, a whole description, from its model, occupancy and control, of the assembled robotic arm can be generated [3]. Here we consider joint and link modules. Each joint module could have more than one degree of freedom and it is constituted by two parts, named proximal part and distal part that are connected by a joint, named P^J_{i-1} in Fig. 1. On the proximal part lies the input connector, while the output connector lies on the distal part. The dashed line represents the axis, around which the rotation of the joint is allowed. Fig. 1 represents a joint module $i-1$ followed by two link modules i and $i+1$. No degrees of freedom are added to the robotic structure by link modules. An input connector and an output connector are present to make the assembly of consecutive modules possible.

2.2. Essentials of the URDF

The URDF is an XML format for describing a robot model, which provides a description of the robot in terms of kinematic, dynamic and visual properties [2]. The file consists of a tree structure alternating link and joint elements. Each joint must be connected to two links, called the parent and child link. Each element presents a sub-tree, where all its own characteristics are specified. In order to be valid, a link element must be inserted at the end of the kinematic chain. The link element in the URDF describes the corresponding robot component from a dynamic and visual point of view, by specifying the mass, the center of mass and the inertia tensor. A shape format or a mesh file are also used for the visualization and collision detection. The joint element characterizes the robot from a kinematic point of view, by presenting the attributes xyz and rpy, which describe the translation and rotation components of joint frame i from the joint frame $i-1$ of the previous element. All the dynamic properties of each link are expressed with respect to its parent joint frame.

[1] SW to URDF Exporter, http://wiki.ros.org/sw_urdf_exporter/.
[2] URDF Homepage, http://wiki.ros.org/urdf/.

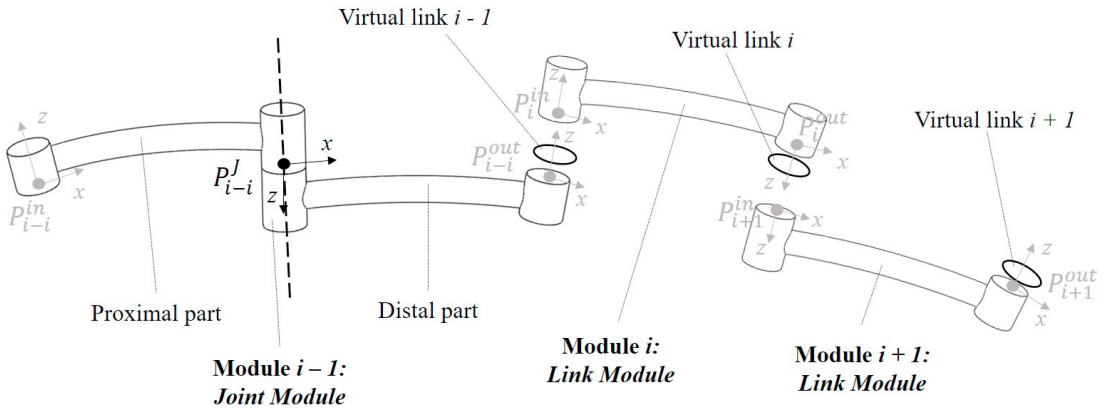


Fig. 1: Schematic representation of an assembly of a joint module and two link modules. Input and output connector frames are represented in grey colour.

The URDF is the standard description format within the ROS libraries, which is an open-source framework for advanced robotics applications [9]. The URDF can be uploaded in ROS simulators and motion planners such as Gazebosim [3] and MoveIt [4]. In this way, visualization and programming of a robot with an URDF is readily possible.

3. Results

3.1. Module's design and established conventions

In this work we propose an algorithm which takes as input the URDF files of the single modules with their desired arrangement and provides the final URDF of the assembled robot as a result. For clarity of subsequent description, we first introduce the representation of the single modules we consider. All the considerations in this work are based on the SolidWorks URDF exporter plugin. This plugin works in such a way that a SolidWorks part is converted into a single URDF link element, while when a SolidWorks assembly is exported, a joint is detected at the connection point between two subsequent parts. By looking at the definition in Subsection 2.1, the input and output connectors are defined for each module. The position and orientation of these connectors are crucial for the correct kinematics definition of the robot, but since they do not add degrees of freedom to the robotic structure, they could be seen as fixed joints. For this reason, it is necessary that during the design, both link and joint modules are modelled as an assembly. For the link element we introduce a virtual link as an auxiliary part of negligible mass and dimensions that will be removed in a further step as described next in Subsection 3.2.

The above mentioned virtual link is necessary for the detection of the output connector. Every time an URDF is obtained, all the properties of the first link element are arranged with respect to the global origin of the CAD software. The connection point between the link itself and the virtual link is the output connector. A sample URDF description of the link module by following these conventions is presented in Listing 1. The dynamic parameters referred to the virtual link are shortened and included between the opening and closing tags (< and / >) for brevity.

For the joint module, the parts involved in the assembly are the proximal part, the distal part, the connection point between them, which is the joint itself and the virtual link, connected by the output connector. A representation of this module as URDF is presented in Listing 2, where the parameters already reported in Listing 1 are here shortened for brevity.

As it can be observed from Listing 2, the URDF could also be written by hand, without the need to export it from a CAD software. The value added by a CAD software is in the possibility to export the mesh files describing each part

[3] Gazebo Homepage, http://gazebosim.org/.
[4] MoveIt Homepage, https://moveit.ros.org/.

Module := link_module

```
<robot name="link_module">
    <link name="link1"> <!-- Start of link -->
        <inertial>
            <origin xyz="0 0 0" rpy="0 0 0" />
            <mass value="0" />
            <inertia ixx="0.0" ixy="0.0" ixz="0.0" iyy="0.0" iyz="0.0" izz="0.0" />
        </inertial>
        <visual>
            <origin xyz="0 0 0" rpy="0 0 0" />
            <geometry>
                <mesh filename="package://my_pkg/meshes/link1.STL"/>
            </geometry>
        </visual>
        <collision>
            <origin xyz="0 0 0" rpy="0 0 0" />
            <geometry>
                <mesh filename="package://my_pkg/meshes/link1.STL"/>
            </geometry>
        </collision>
    </link> <!-- End of link -->
    <joint name="output_connector" type="fixed"> <!-- Start of output_connector -->
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <parent link="link1"/>
        <child link="virtual_link"/>
        <axis xyz="0 0 0"/>
    </joint> <!-- End of output_connector -->
    <link name="virtual_link"> <!-- Start of virtual_link -->
        <inertial/>
        <visual/>
        <collision/>
    </link> <!-- End of virtual_link -->
</robot>
```

Listing 1: Exemplary URDF description of a link module, where its characterizing parameters are set to 0.

from a visual point of view and the estimation of the dynamic parameters such as mass, coordinates of center of mass and inertia tensor. Please note that in case of using SolidWorks for the creation of URDF of modules, it is important to verify whether during the exporting of the meshes, the z axis of the frame for the input connector of each module is directed inwards the input connector's surface, and outwards the module's output connector surface. Consequently, the x and y axes of the input and output frames lie on the plane of the respective connector surface. By denoting as i a general robot module, the result of this convention for the input connector P_i^{in} and for the output connector P_i^{out} can be visualized in Fig. 1. With P_i^J we refer to the frame where the actual joint of the module is located. In the next subsection, the proposed procedure for automatic generation of the URDF file is described.

3.2. Automatic generation of the URDF file

By following the modules' naming defined in Listing 1 and Listing 2, Algorithm 1 describes the procedure implemented for automatic generation of the URDF file of the assembled robot, which requires the URDF files of the set of assembled modules and the order of their intended assembly as inputs. In the resulting XML file as an output, the URDF of the N modules are included. To have at the end a valid robotic chain, the virtual link of each module i is removed and the name of the first link of the successive module $i + 1$ is indicated as the child link of the output connector i. Fig. 1 can be taken as reference for understanding the connection procedure of a joint module with two

Module := joint_module

```
<robot name="joint_module">
    <link name="proximal_part"/> <!-- Block of proximal_part -->
    <joint name="joint1" type="continuous"> <!-- Start of joint -->
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <parent link="proximal_part"/>
        <child link="distal_part"/>
        <axis xyz="0 0 1"/> \\
    </joint> <!-- End of joint -->
    <link name="distal_part"/> <!-- Block of distal_part -->
    <joint name="output_connector" type="fixed"> <!-- Start of output_connector -->
        <origin xyz="0 0 0" rpy="0 0 0"/>
        <parent link="distal_part"/>
        <child link="virtual_link"/>
        <axis xyz="0 0 0"/>
    </joint> <!-- End of output_connector -->
    <link name="virtual_link"/> <!-- Block of virtual_link -->
</robot>
```

Listing 2: Exemplary URDF description of a joint module, where its characterizing parameters are set equal to 0.

Algorithm 1: Synthesis of the robot's URDF from modules.

Output: URDF of the assembled robot (*xml*) into a created ROS package;
Input: modulesFile = [module$_i$.urdf, ..., module$_N$.urdf for module$_i$ ∈ link_module or joint_module];
for $i \leftarrow 1$ *to* N **do**
  copy Mesh File of module_i into final ROS package;
  if *module_i == link_module* **then**
    xml.write(*link_i*);
    xml.write(*output_connector_i*);
  else
    xml.write(*proximal_part_i*);
    xml.write(*joint_i*);
    xml.write(*distal_part_i*);
    xml.write(*output_connector_i*);
  end
  changeChildName;
end
return *xml*

link modules. The virtual links, represented as disks of negligible thickness, of the $i-1$ and i modules are removed from the chain, while the last one of the module $i+1$ is kept to store information about the last output connector. This allows the connection with a further element in the future (e.g., a gripper).

A scheme of the whole approach's implementation is reported in Fig. 2: the first step is the modules' design in SolidWorks considering our proposed convention described in Subsection 3.1, then the model is exported as URDF. At the end, the modules' combination is selected and the final robot can be obtained and visualized. To verify the effectiveness of the approach in simulation, realistic modules have been designed in SolidWorks and exported to URDF. By starting from this set of models, different compositions of these modules have been selected as input to obtain various robots by applying Algorithm 1.

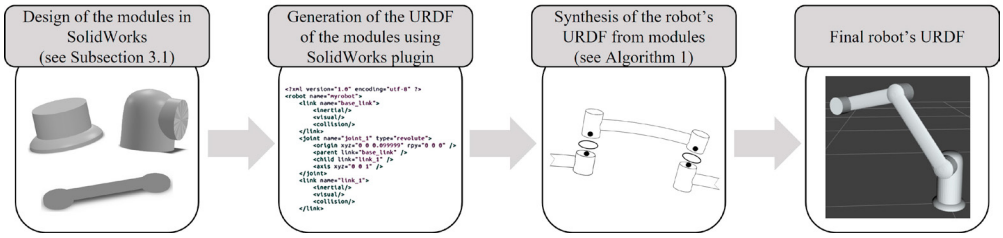
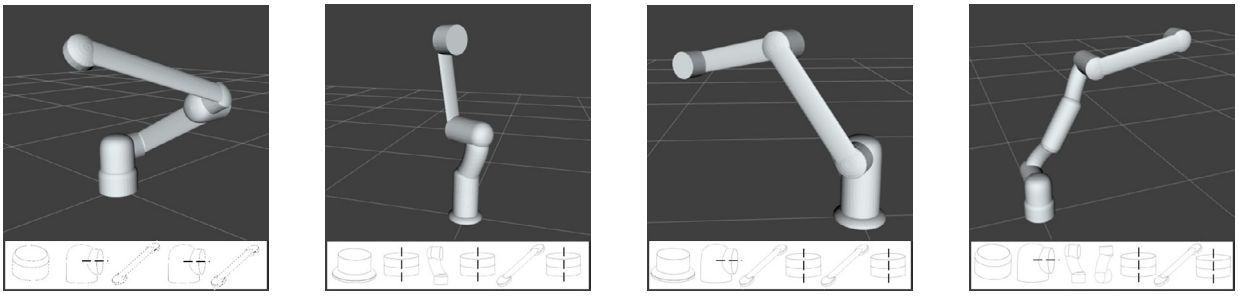


Fig. 2: Illustration of the implementation of the approach starting from the modules designed in SolidWorks, to the URDF exportation, until the algorithm application and the final robot's visualization.



(a) First modular robot. (b) Second modular robot. (c) Third modular robot. (d) Fourth modular robot.

Fig. 3: Different combinations of modules lead to several robotic arms.

Fig. 3 reports four of the many possible assemblies, uploaded into the ROS visualization tool, RVIZ [5]. Under each picture the set of sketches of the modules that has been selected for the obtained assembly is shown.

4. Discussion

This approach ensures a straightforward obtainment of modules described according to URDF from the SolidWorks to URDF exporter plugin. A high customization level can be achieved with several modules' combinations, varying not only their connection order, as demonstrated in Fig. 3, but also their size, material, and shapes. The assembly of single URDF files after the exportation from the CAD software, is ensured by the presence of the virtual links. Indeed, without this choice, it would be not possible to store the information related to the output connector frame of each module. Thanks to this method, modular robots can be easily included into simulation tools and the effort required for changing one or more modules reduces significantly. This work only considers modules that can compose robot arms as [7] does. Extensions would be needed for supporting also the presence of mobile robot bases.

5. Conclusions

This work provides an approach to automatically generate the URDF file of a modular robot manipulator, starting from an extended URDF description of its modules, designed using SolidWorks. This extension, with the inclusion of fixed joints and virtual links for describing the modules, results in a simple yet effective way to enable the automatic generation of the URDF file of a modular robot composition. Although SolidWorks has been exploited for this work, further studies can be carried out to extend the approach to other CAD softwares. Simulations with off-the-shelf tools available in the ROS ecosystem show compatibility with the results of the presented approach and verify its effectiveness.

[5] RVIZ Homepage, http://wiki.ros.org/rviz.

Acknowledgements

This work was supported by the Research Südtirol/Alto Adige grant for the project Reconfigurable Collaborative Agri-Robots (RECOARO) with CUP I52F20000300005.

References

[1] J. Lee, B. Bagheri. and H. Kao "Recent Advances and Trends of Cyber-Physical Systems and Big Data Analytics in Industrial Informatics" *Conference on Industrial Informatics (INDIN)*, 2014.

[2] Y. Liu, X. Ma, L. Shu, G. P. Hancke and A. M. Abu-Mahfouz, "From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges," in *IEEE Transactions on Industrial Informatics*, vol. 17, no. 6, pp. 4322-4334, June 2021.

[3] M. Althoff, A. Giusti, S.B. Liu, A. Pereira "Effortless creation of safe robots from modules through self-programming and self-verification" *Science Robotics*, vol.4, no.31, 2019.

[4] E. Meister, E. Nosov and P. Levi, "Automatic onboard and online modelling of modular and self-reconfigurable robots", 2013 *6th IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, 2013, pp. 91-96.

[5] S. B. Liu and M. Althoff, "Optimizing performance in automation through modular robots," *IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4044-4050.

[6] A. Giusti and M. Althoff, "On-the-Fly Control Design of Modular Robot Manipulators," *IEEE Transactions on Control Systems Technology*, July 2018, vol. 26, no. 4, pp. 1484-1491.

[7] C. Nainer, M. Feder and A. Giusti, "Automatic Generation of Kinematics and Dynamics Model Descriptions for Modular Reconfigurable Robot Manipulators", in *IEEE 17th International Conference on Automation Science and Engineering*, 2021, pp. 45-52.

[8] I.-M. Chen and G. Yang, "Automatic model generation for modular reconfigurable robot dynamics," J. Dyn. Syst., Meas., Control, vol. 120, pp. 346–352, 1998.

[9] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Ng, "ROS: an open-source Robot Operating System", in *Proc. ICRA Open-Source Softw. Workshop*, 2009.

[10] S. Thongnuch and A. Fay, "A practical simulation model generation for virtual commissioning," *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2017, pp. 1077-1082, doi: 10.1109/AIM.2017.8014162.

[11] P. Vyavahare, S. Jayaprakash and K. Bharatia, "Construction of URDF model based on open source robot dog using Gazebo and ROS," *2019 Advances in Science and Engineering Technology International Conferences (ASET)*, 2019, pp. 1-5, doi: 10.1109/ICASET.2019.8714265.