Robot description formats and approaches: Review

1st Mikhail Ivanou
Center for Technologies in Robotics and Mechatronics Components
Innopolis University
Innopolis, Russia
m.ivanov@innopolis.ru

2nd Stanislav Mikhel
Center for Technologies in Robotics and Mechatronics Components
Innopolis University
Innopolis, Russia
s.mikhel@innopolis.ru

3rd Sergei Savin
Center for Technologies in Robotics and Mechatronics Components
Innopolis University
Innopolis, Russia
s.savin@innopolis.ru

2021 International Conference "Nonlinearity, Information and Robotics" (NIR) | 978-1-6654-2407-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/NIR52917.2021.9666120

***Abstract*—Modern robotics represents a collaboration between industry and scientific community, with interdisciplinary research being at its forefront. This leads to wide variation in the requirements for robotic systems, which encourages the development and use of different description formats. Currently, some of the existing formats are gaining wide recognition; at the same time, their individual limitations become more apparent. The goal of this work is to generalize the idea of a robot description format, putting together different approaches to representing a robot, including formats that rely on human-readable and binary data representation, formats tied to specific software, and ones that exist independently, and others. We outline the main advantages and limitations of the existing formats, with the aim to make the following efforts at standardization or development more informed.**

***Index Terms*—Robot description, file format, simulation, URDF, 3D scene**

I. Introduction

Most robot development projects use simulation to test control algorithms and interaction with the external world. To do this, the developer must create a description of both the robot itself and the environment. Gazebo, MuJoCo, CoppeliaSim, PyBullet, and Webots [18] are among the most widely used general-purpose simulators today. Each of them uses its own description format: SDF for Gazebo, MJCF for MuJoCo, wbt files for Webots, and so on. This makes it difficult to exchange models between different research groups. Some robot-specific description formats can be useful for solving particular problems, but cannot be extended to describe complex systems. Other formats allow the addition of new features, but it requires community support. At the moment, software in robotics is not developing as intensively as in other areas of IT and this may take a long time.

On the other hand, Web technologies, 3D animation and visualization, are developing rapidly; these areas solve similar problems and use their own description formats [7]. Unlike robotics, the focus here is on visualization, rather than the accuracy of modeling the dynamics of objects and the perception of the surrounding world, but solutions developed in these areas can be applied in robotics too.

In this paper, we review and compare the existing robot description formats and methods. For this purpose, we define several criteria that, in our opinion, are the most significant in terms of ease of use and further development of description methods. The remainder is organized as follows. Section 2 provides a brief introduction to the robot description. Section 3 defines the set of criteria. Section 4 contains the list of description formats and technologies. Section 5 includes a comparison of the described methods. Section 6 summarizes the main contribution of the paper.

The reported study was funded by RFBR, project number 20-38-70155

II. Robot description

A robot can be described from various points of view: as a graph, as a kinematic structure, as a dynamical system, as an electrical-mechanical system, and so on. In this paper, we consider the description from the point of view of modeling motions and interaction with the external world. In the simplest case, the description format contains the list of links and joints including their types and geometric parameters. Additional elements can be introduced to increase the applicability of the model: inertial parameters, colliding elements, friction model, external world elements, sensors, and others. The full list of such elements, as well as the method of representation, typically depends on the capabilities of the simulator.

A well-thought-out description method can help not only in modeling but also in developing a robot. The simpler the format, the less you need to think about meeting formal requirements and the more attention you can pay to the development itself.

III. Criteria

Any robot description method should provide the ability to model kinematics, dynamics, and contact interaction (collisions) of the robot. These requirements are mandatory. Here we would like to consider other requirements, less obvious but significant from the practical point of view.

- Describing multiple robots: modern modeling systems have to take into account multiple factors and interactions with different objects.
- Support: the system should be continuously developed following the technologies used in the industry.
- Active community: for various reasons, good solutions may remain unknown.

Other parameters are good to consider as well.

- Support for sensors, motors, and controllers: In many cases, it allows finding more reliable simulation results.
- Support for parallel structures: ability to describe both open-loop and closed-loop chains.
- Modular structure: it allows to describe each structural element as a separate module or a file and combine them to describe complex objects. Also, it gives a possibility to reuse components in different robots.
- Code generators and parsers: it is convenient when the programming language allows you to access the elements of the loaded model in the form of class objects or other built-in structures of the language. Another approach is to generate source code based on some description files.
- Convertors: Since different programs or simulators use different description methods, it is convenient to use source format that can be easily converted to the desired program.
- Readability: Human-readable code is easy to understand and modify. On the other hand, such files are larger than in the binary representation, and manually changing the parameters can lead to errors.

This list is not exhaustive, but the presented criteria are sufficient for the analysis from our point of view.

IV. Description approaches

There are two main approaches to the problem of object description. The first one uses some specific type of format. Typically, it is based on any existing markup language like XML. All the description elements have to be defined as elements of the base markup language. The second approach is to provide an interface for building the description file. In this case, the developer has to define the required data structures. After that, the description can be generated and, typically, translated between the different platforms.

A. URDF

The Robot Operating System (ROS) introduces URDF (universal robot description format) [24] framework, and it is widespread today. Its popularity is associated with popularity of ROS itself and the simplicity of the format [3]. The main advantages of URDF are:

- allows to describe robot kinematics, dynamics, colliding elements, and visualization;
- supported with different simulators;
- complex structures can be generated with XACRO templates.

While URDF claims to be "universal," its scope of application is mostly limited to a single robot with a tree-like kinematics structure when the purpose of the project is in the robot's kinematics, dynamics, and visualization only. The main drawbacks of the URDF are the following:

- putting multiple robots together can only be done by combining them (manually or through the use of XACRO) into a single robot element;
- the URDF does not support closed-loop chains;
- sensors are defined in specification but not used in practice;
- the URDF defaults the initial state of a robot to zero joint values. In some cases it can be unfeasible of inconvenient.
- URDF does not provide a way to describe motor model parameters;
- does not support a description of controllers.

These shortcomings were highlighted in [16], and an updated version (URDF2) was proposed, but it seems that the community prefers to use other formats to overcome the problems listed above.

B. SDF

SDF (Simulation Description Format) is intended for describing objects and environment, and it is focused on simulation [11]. Just like URDF, SDF is based on XML. Initially, the format was developed as a part of the Gazebo simulator, but now it can be considered as a separate project that is still growing and developing [12].

SDF has the following advantages [16].

- The project has an active community.
- Support for multiple robots in the scene.
- Support for several types of sensors.
- State of the robots and the world can be stored.
- Support of closed-loop chains.
- C++ API is available.
- It is now fairly independent of the Gazebo simulator.

At the same time, the following disadvantages can be found.

- Grouping of elements is not available.
- Cannot specify hardware parameters.
- Cannot be automatically generated with modern CAD software.

Since Gazebo is the leading simulator in the ROS project, some tools allow to convert URDF into SDF, but not back. However, SDF has several Gazebo-specific parameters, which prevents it from being used as a universal format.

C. SRDF

SRDF (Semantic Robot Description Format) is a representation of semantic information about robots. This format is intended to represent information that is not in the URDF file but can be helpful for a variety of applications [2]. It was proposed as a part of the "MoveIt!" project [17], [22], which is used to solve the inverse kinematics problems for robotic manipulators and calculate the feasible trajectories.

SRDF provides additional concepts such as "group", "chain", "virtual joint", "end-effector" and others. It allows us to divide the kinematics-related problems into sub-problems and describe them more clearly. For example, we can ask the robot to move the end-effector into the desired position and simultaneously avoid collision in some groups of links and joints.

Since the SRDF format is based on XML, the file can be prepared and edited manually. However, there is a tool called MoveIt Setup Assistant that provides the dialog-based interface for generating the description file.

D. SMURF

SMURF (Supplementable, Mostly Universal Robot Format) can be seen as an extension of URDF with additional information, encoded in YAML format [6]. SMURF defines a hierarchy of files that provide information on different aspects of a robot. Originally it was the part of the MARS simulator [5], but the parser was turned into a separate library.

In addition to robot kinematics and dynamics, SMURF defines such elements as sensors, motors definition, and simulation parameters. In general, the format allows the user to specify any information related to any part of the robot or the model as a whole, using the YAML syntax. The C++ parser can embedded into an application to extract the required data from the SMURF file.

The format is based on URDF, which makes it compatible with the ROS framework. The Phobos plugin allows using free CAD Blender for the SMURF model visualization.

E. CoppeliaSim files

CoppeliaSim is a popular robot simulator. It uses two types of binary files, one for models and another for scenes. All operations with scenes and models can be done by graphical tools. XML-based files are supported too, but they are used either as a version-independent data storage or to prepare the model for another program [14].

CoppeliaSim description includes the full image of the scene and models that contain all required information, such as geometrical and dynamical parameters, types and positions of sensors, motor settings, object location in the world, mesh form, joint states. Additional plugins allow the simulator to import URDF and SDF files and export scenes of simulation via glTF format.

F. MJCF

MJCF is an XML-based file format that describes a model in MuJoCo simulator (Multi-Joint dynamics with Contact) [26]. This file format contains not only a description of the robot structure but, among other things, sensors, motors, and contacts. For runtime computation, the model is compiled into a low-level optimized data structure mjModel and can be saved in a binary MJB file.

MJCF provides more elements and attributes than URDF format [8], however only a few of them need to be defined explicitly by the user while others can take default values. According to the developers, this makes MJCF files on average shorter and more readable than URDF files defining the same robots. The disadvantage of this approach may be the unexpected behavior of the model due to the influence of implicit parameters. MJCF can include links to other files, allowing a modular description format. It should be noted that URDF can be used in MuJoCo as well.

G. WBT

Webots robot simulator [23] provides a graphical method of world definition. All elements are represented in the form of tree nodes. The models are stored in WBT text files. On the one hand, WBT is a subset of the nodes and fields specified by the VRML97 (The Virtual Reality Modeling Language) standard [1]. On the other hand, some elements were extended for the robot modeling.

The list of nodes is not limited to the predefined elements, user can add own nodes using the PROTO mechanism [13]. Once defined, PROTO nodes may be used exactly like built-in elements. Thus, users can build and reuse complex objects.

H. COLLADA and OpenRAVE's extensions

COLLADA (COLLAborative Design Activity) is the XML-based file format that helps to transport 3D assets between different applications. It provides encoding for visual scenes including geometry, shaders, and effects, as well as descriptions of physics, animation, and kinematics. In 2013 Collada 1.5 has been published as an official ISO standard to provide a reliable, long-term international standard for 3D asset authoring and interchange [4].

COLLADA has not been developed for robotic applications. However, it was shown how the format could be modified to work with robots and translate scenes between different simulators [20]. The authors identified two main features of COLLADA:

- assumption that the information format is continuously evolving;
- ability to specify the same information using different standards.

Several of the currently used CAD systems can export files in the COLLADA format. The main drawback is the low readability from a human point of view. Besides, there is no support for actuators/transmission specifications by default, but it can be solved using the OpenRAVE robot-specific extensions [19].

I. glTF and RDTF

glTF (Graphics Language Transmission Format) is a file format for three-dimensional scenes and models focused on efficient transmission and loading by engines and applications. The standard is open; it supports 3D model geometry, appearance, scene graph hierarchy, and animation [9].

The structure of the format is strictly hierarchical. glTF defines two types of representation: JSON-based and binary. The first can be self-contained or may reference external files (textual or binary), the second is self-contained. Often JSON-based file contains scene structure, while the binary file store all the required scene data.

glTF is young enough but has an extensive ecosystem of supported applications and technologies, particularly OpenGL and WebGL projects.

Extention of glTF for robotic applications was proposed in paper [21]. The authors defined a set of fields with robot-specific elements, called RDTF (Robot Description Transmission Format), put it into the "extras" element, and defined algorithms for data conversations. This approach can work with the robot models using widely used tools for editing and visualization.

TABLE I
COMPARISON OF THE ROBOT DESCRIPTION METHODS.

	SDF	URDF	SRDF	SMURF	Coppelia Sim files	COLLADA (openRAVE)	MJCF	WBT	glTF (RDTF)	Phobos	Nvidia Builder	USD
Multiple robots	+	-	-	-	+	-	+	+	+	-	+	+
Scene description	+	-	-	-	+	-	+	+	+	-	+	+
Wide use	-	+	-	-	-	+	-	-	+	-	-	-
Modular	+	+	+	+	-	-	+	+	+	n/a	n/a	+
Human readable	+	+	+	+	-	+	+	+	+/-	n/a	n/a	+
Sensors	+	-	+	+	+	+	+	+	+	+	-	+
Closed-loop chains	+	-	-	-	+	+	+	+	-	-	-	-
Syntax	XML	XML	XML	XML+ YAML	bin	XML	XML	VRML	JSON+ bin	Inter face	Inter face	Cus tom

J. Phobos plugin

Phobos is an add-on for the 3D modeling system Blender that enables the creation of WYSIWYG (What You See Is What You Get) robot models. The add-on exports formats such as URDF, SDF, SMURF, and common mesh formats (STL, OBJ, COLLADA). It allows the definition of kinematic parameters and visualizes different model components. Also, it contains numerous editing tools: auto-generation of collision objects, auto-generation of inertia tensors from mass and shape, batch editing of object properties, calculation of merged inertia for complex links [25]

K. Isaac Sim and Robot Builder

Another system that can load URDF models is Isaac Sim, a simulator developed by Nvidia. It uses the RobotBuilder plugin that loads the model into the Unreal Engine. After that, the robot can be configured and spawn using a set of Blueprint nodes, the scripting system based on the concept of using a node-based interface to create gameplay elements from within Unreal Editor. In addition to blueprints, robot configurations can also be defined in JSON. This format can override the defaults when creating a robot from URDF or combining several robots in one scene and with each other.

As of 2020, Nvidia no longer supports UE4-based Isaac Sim. Now the simulator is built on NVIDIA Omniverse and uses USD.

L. USD

Pixar has created USD (Universal Scene Description), the framework for reliable and scalable data exchange of 3D computer graphics. USD supports huge 3D scenes that can be composed of multiple elementary assets [15].

Initially, the format and the software were created for animation, but it was adopted for the robotics applications by Nvidia's Omniverse Isaac Sim. Using the format's effective layering and reference structure Isaac Sim provides the necessary functions for building virtual robotic worlds and conduct experiments.

V. DISCUSSION

Comparison of the different robot description methods can be found in Table I. Some criteria were omitted, as it is difficult to talk about their full support or absence. For example, on the simulator side the text format requires tools for text analysis, while on the user side it is desirable to have generation tools that alleviate unnecessary copying and prevent data entry errors associated with manual data manipulation. Currently, there are various open source programs that partially solve these problems; however, due to the limited support and stability issues they might not be fully usable in professional settings. There is no ideal approach to describe robots, but such formats like SDF, MJCF, WBT, glTF, USD are close.

The most straightforward way to define the robot and environment is the WYSIWYG approach, for example, using a CAD system to convert the result to any format that the simulator needs. However, this method requires special tools and work skills. Therefore, at least for now, the XML-based description seems more straightforward. SDF is not the most widely used approach; besides, it can be easily obtained from well-known URDF files to support multiple robots, even without ROS [10].

In recent years, a lot of 3D scene formats have been introduced. When working with single robot or simple environment, it is enough to use well-known old methods. Movies, games, VR and other 3D visualization directions are developing much faster than robotics, and they are focused on reliable and practical work with vast scenes of objects. Furthermore, their results can help in environment simulation for robotics experiments that were previously impossible.

VI. CONCLUSIONS

In this paper, we described several robot description formats and approaches that are used or can be used in the future by robotics researchers and developers. We also defined a list of criteria significant from the practical point of view and compared it.

We expect that the ideal robot description format can define scenes, multiple robots and their components (sensors, actuators), complex kinematic structures (loops), and their states, allows to dividing into sub-modules. It should be human-readable but generated with WYSIWYG tools and easily translated between different simulators and visualization tools. Moreover, it should be a widely used standard.

Several standards are close to this description in one way or another, but each of them is used in its ecosystem. Thanks to supporting from different programs, URDF is still the most "universal" format despite its limitations.

References

[1] The virtual reality modeling language (1997), https://tecfa.unige.ch/guides/vrml/vrml97/spec/
[2] Semantic robot description format review (2011), http://wiki.ros.org/srdf/review
[3] Xml robot description format (urdf) (2012), http://wiki.ros.org/urdf/XML/model
[4] Khronos collada now recognized as iso standard (2013), https://www.khronos.org/news/press/khronos-collada-now-recognized-as-iso-standard
[5] Mars (2017), https://github.com/rock-simulation/mars/wiki
[6] A parser for smurf files (2017), https://github.com/rock-simulation/smurf_parser
[7] Top 10 3d modeling & animation file formats in 2020 (2020), https://winbizsolutionsindia.com/3d-file-formats/
[8] Chapter 5: Xml reference (2021), http://mujoco.org/book/XMLreference.html
[9] gltf overview (2021), https://www.khronos.org/gltf/
[10] Script to convert robot models (2021), https://github.com/l1va/robot_shop
[11] Sdformat (2021), http://sdformat.org/
[12] Simulation description format parser and description files (2021), https://github.com/osrf/sdformat
[13] Webots reference manual (2021), https://cyberbotics.com/doc/reference/proto
[14] Xml formats (2021), https://www.coppeliarobotics.com/helpFiles/en/xmlFormat.htm
[15] Blevins, A., Murray, M.: Zero to usd in 80 days: transitioning feature production to universal scene description at dreamworks. In: ACM SIGGRAPH 2018 Talks, pp. 1–2 (2018)
[16] Chitta, S.: Urdf 2.0: Update the ros urdf format (2021), http://sachinchitta.github.io/urdf2/
[17] Chitta, S., Sucan, I., Cousins, S.: Moveit![ros topics]. IEEE Robotics & Automation Magazine **19**(1), 18–19 (2012)
[18] Collins, J., Chand, S., Vanderkop, A., Howard, D.: A review of physics simulators for robotic applications. IEEE Access (2021)
[19] Diankov, R., Kuffner, J.: Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 **79** (2008)
[20] Diankov, R., Ueda, R., Okada, K., Saito, H.: Collada: An open standard for robot file formats. In: Proceedings of the 29th Annual Conference of the Robotics Society of Japan, AC2Q1–5 (2011)
[21] Izawa, R., Koga, M.: An extension of gltf for robot description. In: 2019 19th International Conference on Control, Automation and Systems (ICCAS). pp. 1010–1014. IEEE (2019)
[22] Kunze, L., Roehm, T., Beetz, M.: Towards semantic robot description languages. In: 2011 IEEE International Conference on Robotics and Automation. pp. 5589–5595. IEEE (2011)
[23] Michel, O.: Cyberbotics ltd. webots™: professional mobile robot simulation (2004)
[24] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y., et al.: Ros: an open-source robot operating system. In: ICRA workshop on open source software. vol. 3, p. 5. Kobe, Japan (2009)
[25] von Szadkowski, K., Reichel, S.: Phobos: A tool for creating complex robot models. Journal of Open Source Software **5**(45), 1326 (2020)
[26] Todorov, E., Tom, E., Yuval, T.: Mujoco: A physics engine for model-based control (2012)