FP7-600716

Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

D1.1
Enhanced iCub simulator for whole-body contact simulation

Editor(s)	Vincent Padois[1,2]
Responsible Partner	UPMC
Affiliations	[1] Sorbonne Universités, UPMC Paris 06, UMR 7222, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005, Paris, France. [2] CNRS, UMR 7222, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005, Paris, France.
Status-Version:	Final-1.0
Date:	Feb. 28, 2014
EC Distribution:	Consortium
Project Number:	600716
Project Title:	Whole-Body Compliant Dynamical Contacts in Cognitive Humanoids

Title of Deliverable:	Enhanced iCub simulator for whole-body contact simulation
Date of delivery to the EC:	28/2/2014

Workpackage responsible for the Deliverable	WP1
Editor(s):	Vincent Padois
Contributor(s):	Sovannara Hak, Serena Ivaldi, Mingxing Liu, Vincent Padois (UPMC) / Andrea Del Prete, Francesco Nori, Daniele Pucci, Francesco Romano, Silvio Traversao (IIT) / Leon Žlajpah (JSI)
Reviewer(s):	
Approved by:	All Partners

Abstract	The work described in this deliverable is part of WP1 and aims at providing the CoDyCo consortium with a shared framework for the simulation of humanoid robots and/or digital humans involved in whole-body and multi-contact activities. In order to reach this goal, several activities have been led in parallel. They are described in this deliverable.
Keyword List:	dynamics simulators, humanoid robots, digital human, URDF

Document Revision History

Version	Date	Description	Author
v. 0.1	Feb. 20, 2014	Initial draft	Vincent Padois
v. 0.5	Feb. 25, 2014	Intermediate version	Vincent Padois
v. 0.9	Feb. 27, 2014	Final version	Vincent Padois
v. 1.0	Feb. 28, 2014	Proofread version	Francesco Nori

Table of Contents

Index of Figures

1 Introduction

The work described in this deliverable is part of WP1 and aims at providing the CoDyCo consortium with a shared framework for the simulation of humanoid robots and/or digital humans involved in whole-body and multi-contact activities. In order to reach this goal, several activities have been led in parallel. They are described in this deliverable as follows. In Section 2, the definition of the requirements for the simulation framework is provided. In Section 3, the objectives of a survey of the existing simulators for robotics are presented. In Section 4, iCubsim, the historical iCub simulator developed at IIT, is described and two alternative solutions, better suited for the simulation of whole-body motions in contact, are introduced. The results of the comparison between these two simulators and a real iCub performing a free-falling task are provided in Section 5. The work achieved for this deliverable is summarized in the Conclusion and some perspectives are given. Appendix A contains the survey paper on simulators submitted to the IEEE Robotics and Automation Magazine. Appendix B contains the manual of the digital human URDF model generator developed at JSI.

2 Requirements for enhanced iCub simulator for whole-body contact simulation

In this section, the definition of the requirements for the simulation framework is provided.

2.1 Motivations

With the progress of powerful computers enabling fast computations, dynamics simulation in robotics is no longer expected to be an offline computational tool. It is used to rapidly prototype controllers, evaluate robots design, simulate virtual sensors, provide reduced models for model predictive controllers, supply with an architecture for real robot control, and so on.

This is especially true in the framework of the CoDyCo project where each technical work package can benefit from an efficient, modular dynamics simulator. Such a simulator is for example useful:

- in WP2 to evaluate the validity of dimensionally reduced models of human whole-body motion in contact using simulated digital humans;
- in WP3 to rapidly prototype and evaluate the whole-body reactive controllers and potentially provide computationally efficient models for model predictive controllers;
- in WP4 to bootstrap the learning algorithms without requiring the use of real robots in the first stage of the learning process;
- in WP5 to extensively test the various validation scenarii before running them on the real robots.

2.2 Critical features

Dynamics simulators for robotics have more strict requirements than the ones used for animating virtual characters, where time, computational burden and physical reality can be less constraining. In entertainment (e.g. video-games), infeasible forces may not be a problem since the laws of physics can be violated. In bio/mechanical studies, simulators can be used offline to analyse or synthesize behaviours. Although the field of dynamics modelling and simulation has matured over the last decades [1, 2, 3], the growing need to control whole-body movements of complex structures, such as humanoids, raises additional challenges to simulators for robotics:

1. numerical stability, which strongly restricts the use of simulations in real-time control settings [4, 5];
2. the capability to be used as predictive engines in real-time control loops [6], which requires the ability to be extremely fast in computing the dynamics and the guarantee for the solvers to converge to physically feasible solutions upon a certain time [7];
3. the simulation of rigid and soft bodies in contact with rigid and compliant environments [8, 9]: the inaccurate computation of contact forces between bodies may result in unrealistic contacts or physically infeasible contact forces (this issue has been particularly evident in the virtual phase of the Darpa Robotics Challenge - DRC);
4. the capability to model and simulate new types of actuation systems, such as variable impedance or soft actuators [10], and different types of contacts, for example with deformable materials, compliant and soft surfaces [11].

Modularity is also a critical feature. Indeed, depending on the use made of the simulator all components such as the 3D graphics display, the graphical user interface, the interfaces with input devices (keyboard, space mouse,...), the physics core, the controller, etc. may not be required: a modular, component-based, software architecture can reduce the non-required computational load. It can also permit the use of different solutions for a given component such as the physics core, the 3D graphics display or the controller. Finally, if components are glued together using a middle-ware such as YARP [12], ROS [13] or OROCOS [14], the integration of a controller prototyped in simulation on the real robot can be largely simplified as the way to access (basically set control modes, send control inputs and get sensors feedback) to the simulated and real robot can be the same.

Finally, the robotics community urges for standardized software tools and particularly open source software. The benefit of open-source does not only lie in the community that can grow around the software, developing new tools, improving its quality and avoiding to "re-invent the wheel" at each time, but also in checking its efficiency and robustness on real platforms (which is expensive).

2.3 Conclusions

Within the framework of the CoDyCo project, a modular, component-based dynamics simulation software providing numerically stable, computationally efficient and physically consistent

simulations of whole-body virtual human(oid) systems in contact with rigid or soft environments is required.

3 Survey of existing robotics simulators

There is a growing number of tools for dynamics simulation, ranging from dynamics solver libraries to systems simulation software, provided through either open or closed source code solutions, each more or less tailored to their expected domains of application.

The spectrum of robotics applications being large and expanding, it is necessary for the developers to get feedback about the users' needs, and for the researchers to be aware of the available tools and have the elements to ponder which of the available tools is the best for their research. Most middle-ware for robotics (ROS, YARP, OROCOS, Player, etc.) are already open-source, some also cross-platform. This makes it possible to produce interesting performance comparisons that can help the roboticists to pick the best middle-ware for their needs [15]. Similar ideas (open-source and cross-platform compatibility) should be used to compare dynamics models and simulators. For example, an interesting evaluation and performance comparison of contact modelling algorithms was presented in [4, 5].

As a complement to quantitative comparisons, a useful element of evaluation (often neglected) is user feedback. What do users really think of the software they use for simulation? Would they suggest it? What is their experience in their particular use case? It is believed that user feedback may be useful to avoid time-consuming tuning and inappropriate choices of software to end-users. It could point a researcher to a community that is actively using the tool and that is sharing the same concern: for example, it is likely that people simulating flying robots have different needs than those simulating wheeled robots or those controlling bipeds. Furthermore, user feedback can provide useful suggestions to the developers community about the things that matter the most to users in simulation.

With this goal in mind, an online survey about the use of dynamical simulation in robotics has been created[1]. The survey is divided into four parts: general information about the user, user experience with dynamics simulation in general, user experience with one tool of his choice, technical questions and subjective evaluation about the selected tool. The survey has been advertised on the main robotics mailing lists (e.g., euron-dist, robotics-worldwide) as well as in other mailing lists of correlated disciplines (e.g. comp-neuro), and kept open for approximately one month.

The paper in Appendix A, submitted to the IEEE Robotics and Automation Magazine, summarizes the analysis of the users' answers. A descriptive sheet for the most relevant software tools, for the reader's interest, is also provided. For the complete analysis of the simulators survey, the reader is referred to the extended version of the report[2].

[1] Online survey: `http://goo.gl/Tmyf5A`
[2] http://arxiv.org/abs/1402.7050

4 Proposed technical solutions

In this section, the historical dynamics simulator dedicated to iCub is briefly presented (the proposed description is based on [16]. Given its limitations, two partners (IIT and UPMC) have proposed alternatives solutions: one based on Gazebo and the other one based on XDE. The main features of these two solutions are described. The section ends with the description of the URDF digital human generator developed by JSI.

4.1 Historical solution: the ODE iCub simulator

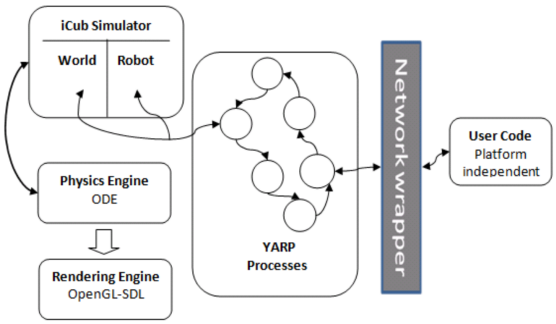
The first iCub simulator has been initially designed by Vadim Tikhanoff to reproduce, as accurately as possible, the physics and the dynamics of the iCub robot and its environment. The simulated iCub robot is composed of multiple rigid bodies connected via joint structures. It has been constructed collecting data directly from the robot design specifications in order to achieve an exact replication (e.g. height, mass, Degrees of Freedom) of the first iCub prototype developed at the IIT within the framework of the RobotCub EU project [17]. The environment parameters on gravity, objects mass, friction and joints are based on estimated environment conditions.

This simulator has been created using open source libraries in order to make it possible to distribute the simulator freely to any researcher without requesting the purchase of restricted or expensive proprietary licenses. Although the proposed iCub simulator is not the only open source robotics platform, it is one of the few that attempts to create a 3D dynamics robot environment capable of recreating complex worlds and fully based on non-proprietary open source libraries.

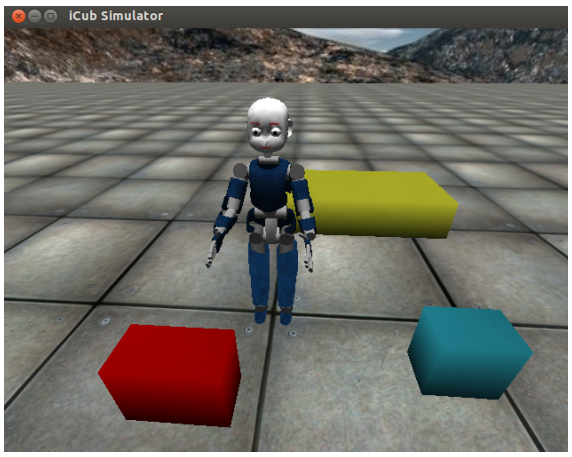
The physics core uses ODE (Open Dynamics Engine) [18] for simulating rigid bodies and detecting collisions algorithms to compute the physical interaction with objects. Rendering is performed using OpenGL combined to SDL (Simple Directmedia Layer) [19], an open source cross-platform multimedia library. As the aim was to create an exact replica of the physical iCub robot, the software infrastructure and inter-process communication which have been used are similar to those used to control the physical robot. iCub uses YARP (Yet Another Robot Platform) [12] as its software architecture / middle-ware. YARP is an open-source software tool for applications that are real-time, computationally intensive, and involve interfacing with diverse and changing hardware. The simulator and the actual robot have the same interface either when viewed via the device API or across network and are interchangeable from a user perspective. A global view of the ODE iCub simulator software architecture is provided in Fig. 1a. A view of the simulator is given in Fig. 1b.

4.2 iCubsim with Gazebo

One of the shortcomings of the ODE iCub simulator is the way ODE represents rigid-body structures: it represents joints as constraints between bodies. A second class of physics core libraries, such as the ones used in XDE [20] and OpenHRP [21], makes use of parametrized rigid-body dynamics representations, where joints are simply part of the robotics structure.



(a) Global view of the software architecture [16].



(b) A view of the ODE iCubSim simulator.

Figure 1: Software architecture and view of the ODE iCub simulator.

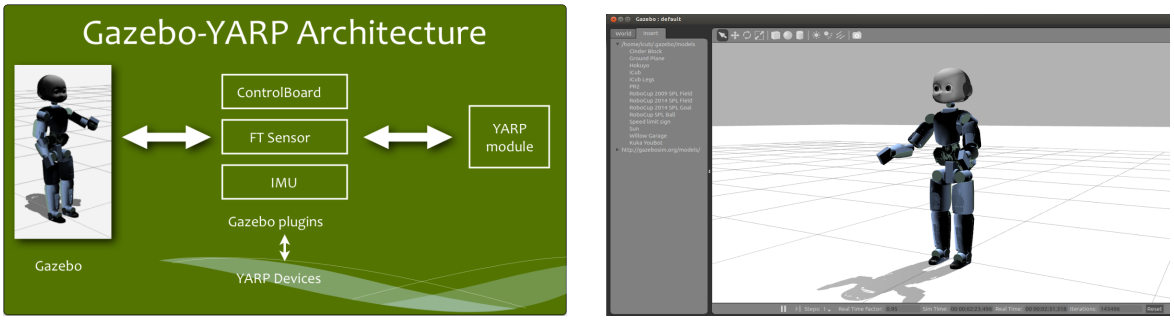
These two classes determine not only the way forward/inverse dynamics are computed (the second class benefits from the straightforward computation of quantities useful in robotics, such as Jacobians, mass matrices etc.), but most importantly the way contact forces are computed. The first class considers contacts forces as bilateral/unilateral constraints, which are added to the list of constraints used to describe the joints; then the same solver is used to find the forces for the global system, including contacts and joints. In the second class, on the contrary, only constraints from the contacts are solved, which notably simplifies the problem. This leads to more stable and physically consistent numerical integration results.

In order to benefit from alternative physics core libraries and from a widely used, open-source, modular simulation framework, the IIT has developed a new version of its iCub simulator based on Gazebo [22]. Gazebo is a multi-robot simulator, developed by the Open-Source Robotics Foundation. It is the official software tool for the Darpa Robotics Challenge and it can rely on Bulletphysics [23] as the physics simulation core. The latest version of Bullet (2.82) relies on efficient dynamics computation algorithms developed by R. Featherstone [24] as well as on the state of the art Mixed Linear Complementarity Problems contact solver developed at INRIA [25]. A schematic view of the Gazebo iCub simulator software architecture is provided in Fig. 2a. A view of the simulator is given in Fig. 2b.

4.3 iCub in XDE

Based on its past experience with the Arboris-Python[3] dynamics simulator [26] and on its long lasting collaboration with the CEA LIST, UPMC has chosen to explore the possibilities offered by the XDE modular simulation framework [20].

[3]This simulator was first developed in Matlab by A. Micaelli, researcher at CEA, and then in Python by S. Barthélémy and J. Salini at UPMC.



(a) Schematic view of the software architecture. (b) A view of the Gazebo iCubSim simulator.

Figure 2: Software architecture and view of the Gazebo iCub simulator.

XDE is a modular, extensible, component-based simulation framework based on the Orocos [14] robotics middle-ware and primarily dedicated to interactive simulation and virtual reality applications. XDE is based upon a physics simulation kernel that handles rigid and deformable bodies, in particular cables, multi-body systems with kinematic constraints and intermittent contacts, fluids: liquid, gas or smoke. XDE is equipped with two different proximity computing engines to achieve a precision guaranteed collision detection between objects, even with complex industrial models produced by CAD tools. XDE is based upon widely accepted technologies like Orocos RTT, Python, C++, Ogre, OpenMP, CUDA. It comes with a rigid-body model computation component and its functionalities come with Python wrappers facilitating the rapid prototyping of code.

Based on this framework and on the work of Joseph Salini [27] on whole-body controllers for humanoid robots, a model of iCub in XDE has been implemented and several simulations including iCub in whole-body and multi-contact situations have been performed. A view of the simulator including iCub in a whole-body multi-contact situation is given in Fig. 3.

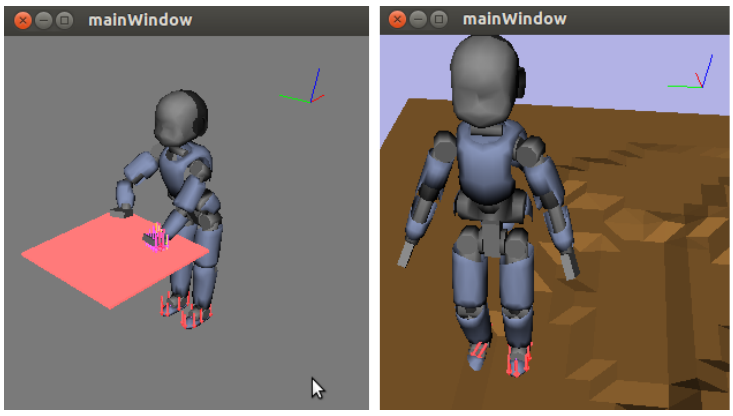


Figure 3: A view of the XDE iCub simulator.

4.4 Digital human URDF file generator

Within the framework of WP2, WP3 and WP4, simulating digital humans using the iCub dynamics simulator (especially in the case where the human-robot interaction is studied) is required. In order to easily generate digital humans of various heights and weights, XDE includes a parametrized digital human model with 45 "actuated" degrees of freedom among which 6 are located in the back. A view of four instances of the same parametrized mannequin with different heights and weights is provided by Fig. 4.

In order to provide a way to generate URDF [28] models of digital mannequins independently from a specific simulator, JSI has developed a software for generating instances of a parametrized digital human (similar to the one present in XDE) as well as to edit the detailed parameters of an existing instance. The user manual of this software is provided in Appendix B.

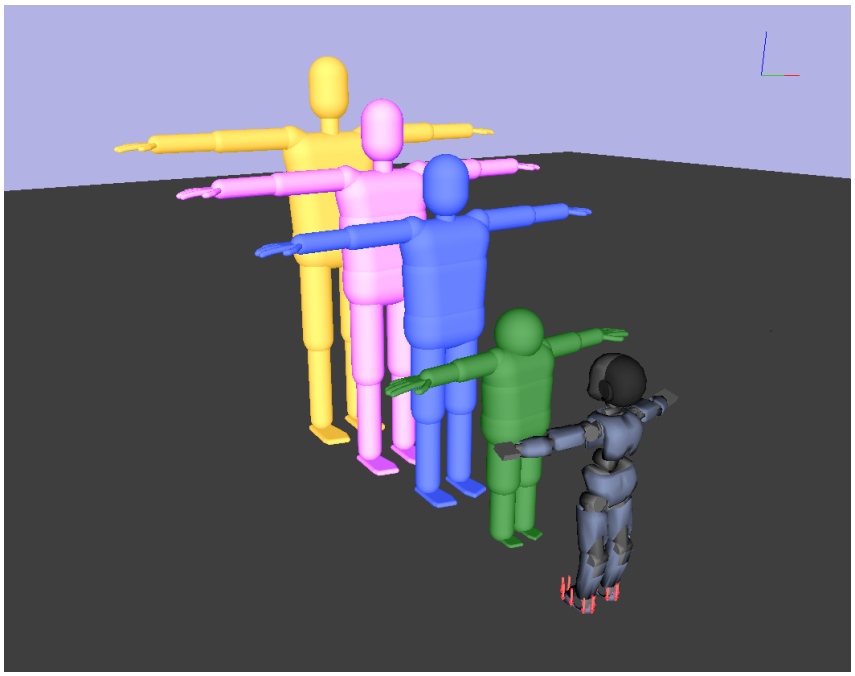


Figure 4: iCub and four instances of the same parametrized mannequin with different heights and weights in XDE.

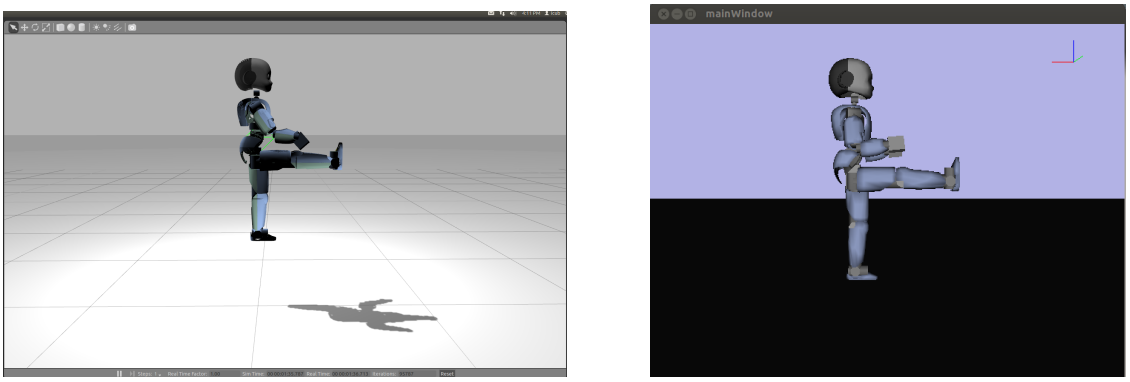
5 Assessment of the proposed simulation solutions

This section describes some preliminary work aiming at quantitatively evaluating the accuracy of the rigid body simulation of the iCub robot in the Gazebo [22] and XDE [20] simulators. In order to perform this comparison, an initial experiment on a real iCub robot is performed. This experiment is used to identify a reasonably accurate model of the friction acting in the real robot so that this model can complete the pre-existing rigid-body model of the robot used for simulation. Friction being a major component of the torques acting on the robot at

low velocities, this first phase is needed to ensure that the comparison between the physical and virtual world robot is meaningful. Once this fairly accurate model is obtained, the second stage consists in performing similar experiments on the real robot as well as on its XDE and Gazebo simulated counterparts.

5.1 Methodology

The experiment chosen to perform the comparison consists in a free-fall of the right leg of iCub, starting from of a configuration where the leg is stretched horizontally as illustrated by Fig 5a and Fig. 5b respectively in the Gazebo and XDE simulators. A shared URDF model of the iCub robot is used in both simulators.



(a) Gazebo view. (b) XDE view.

Figure 5: iCub Free-falling right leg experiment: initial posture.

While all the dynamics parameters of the robot are computed based on the robot CAD model as well as on the various components data-sheets, a friction model is estimated using a software module[4] developed at IIT. The retained friction model $\Gamma_{f,i}$ for joint i can be written as follows

$$\Gamma_{f,i} = (k_{vp}s(\dot{q}) + k_{vn}s(-\dot{q}))\dot{q} + (k_{cp}s(\dot{q}) + k_{cn}s(-\dot{q}))\text{sign}(\dot{q}), \quad (1)$$

where $k_{vp}, k_{cp}, k_{vn}, k_{cn}$ are respectively the viscous and Coulomb friction coefficients for positive joint velocities and the viscous and Coulomb friction coefficients for negative joint velocities. $\dot{q}$ is the joint velocity, $s(x)$ is the step function (1 for $x > 0$, 0 otherwise) and $\text{sign}(x)$ is the sign function (1 for $x > 0$, -1 for $x < 0$, 0 for $x = 0$).

This model ignores the Stribeck friction regime acting at very low velocities and generating stick-slip behaviours. Still, it can reasonably be used in many applications and presents the advantage of begin linear with respect to the friction coefficients, thus making it rather straightforward to identify using least-squares identification techniques.

Two models have been identified: one including both viscous and Coulomb friction components and one including only a hand-tuned/adapted viscous friction component. This

[4] https://github.com/robotology/codyco/tree/master/src/modules/motorFrictionIdentification

hand-tuned coefficient is based on an inverted pendulum model and the obtained value accounts for the mass distribution of the bodies and the observed velocity of the joints on the real robot. Table 1 summarizes the values obtained by IIT on one of their iCub robot.

Joint \ Coefficient	Viscous [N.m.s]	Coulomb [N.m]	Adapted viscous [N.m.s]
Hip	$k_{vp} = 0.69212$ $k_{vn} = 0.68809$	$k_{cp} = 2.64804$ $k_{cn} = 1.78945$	$k'_v = 93.51401$
Knee	$k_{vp} = 0.57051$ $k_{vn} = 0.57482$	$k_{cp} = 2.00711$ $k_{cn} = 3.21653$	$k'_v = 19.84783$

Table 1: Identified friction coefficient for the knee and hip of the iCub robot.

Based on these models and on the real robot experiment, different tests have been performed in both the Gazebo and XDE simulators. It has to be noticed that, both in XDE and Gazebo, viscous friction is intriniscally part of the description of a joint and can thus be integrated implicitly. It is not the case of Coulomb friction. Also, the viscous + Coulomb model exhibits extremely low frictions and thus the retained experiments only compares the real robot motion to the simulated one using the adapted friction model in both simulators.

	Real robot	Gazebo	XDE
Real world friction	A_1	-	-
Adapted viscous friction model	-	B_2	B_3

Table 2: Tests performed in the Gazebo and XDE simulators.

The joint trajectories of the hip and knee are recorded for comparison. These results are presented in the next section.

5.2 Results

Figures 6 and 7 provide plots of the evolution of the hip and knee angle over time in the cases described by Table 2.

The first observation from these curves is the large amount of friction faced by the leg joints of the robot. This is a known problem, especially by control people, in most humanoid robots. iCub is similar to them in that sense.

The second observation is related to the similarity of the results obtained using both XDE and Gazebo. Some minor differences can be observed but, in this experiment, it is impossible to conclude whether one simulator is more accurate than the other. Indeed, in the process of obtaining these results, IIT and UPMC realized the difficulty to share common robots

models, despite the existence of the URDF standard description format. The small difference between the two simulators' results are likely to be due to a mismatch between the used models.

Finally, simulating viscous friction is not enough for accurate simulations. This really emphasizes the importance of identification in order to properly close the gap between the real world robot and its simulated counterpart. This is an on-going effort in CoDyCo, especially in WP1, WP3 and WP4.

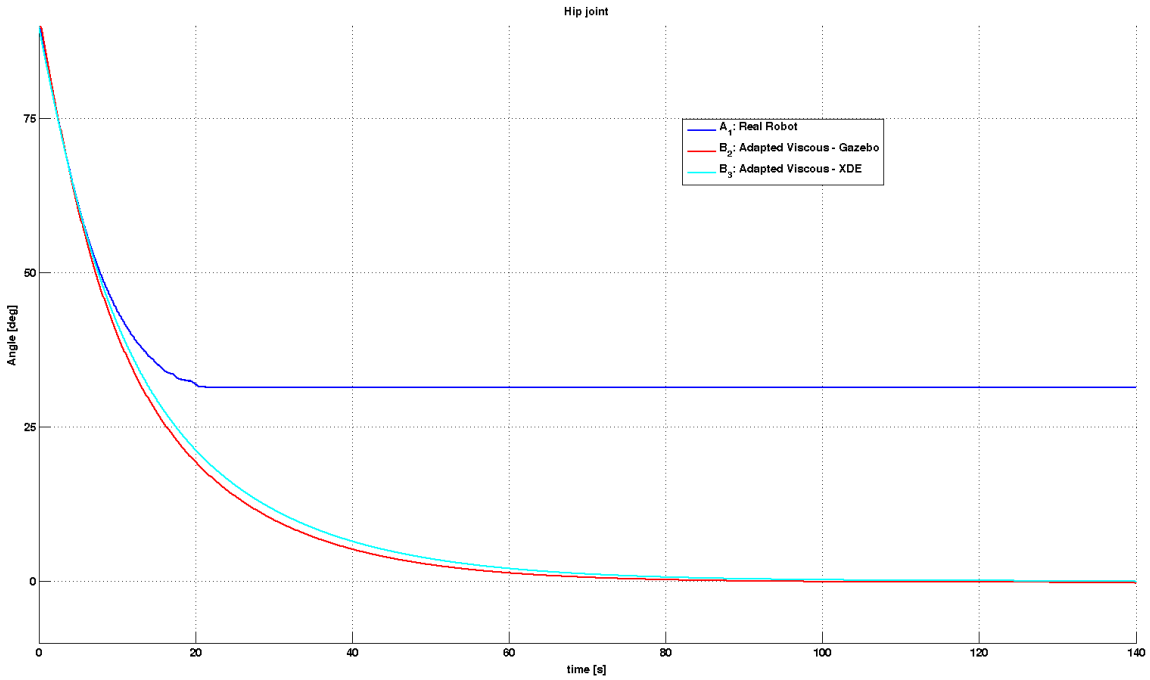


Figure 6: Evolution of the hip angle.

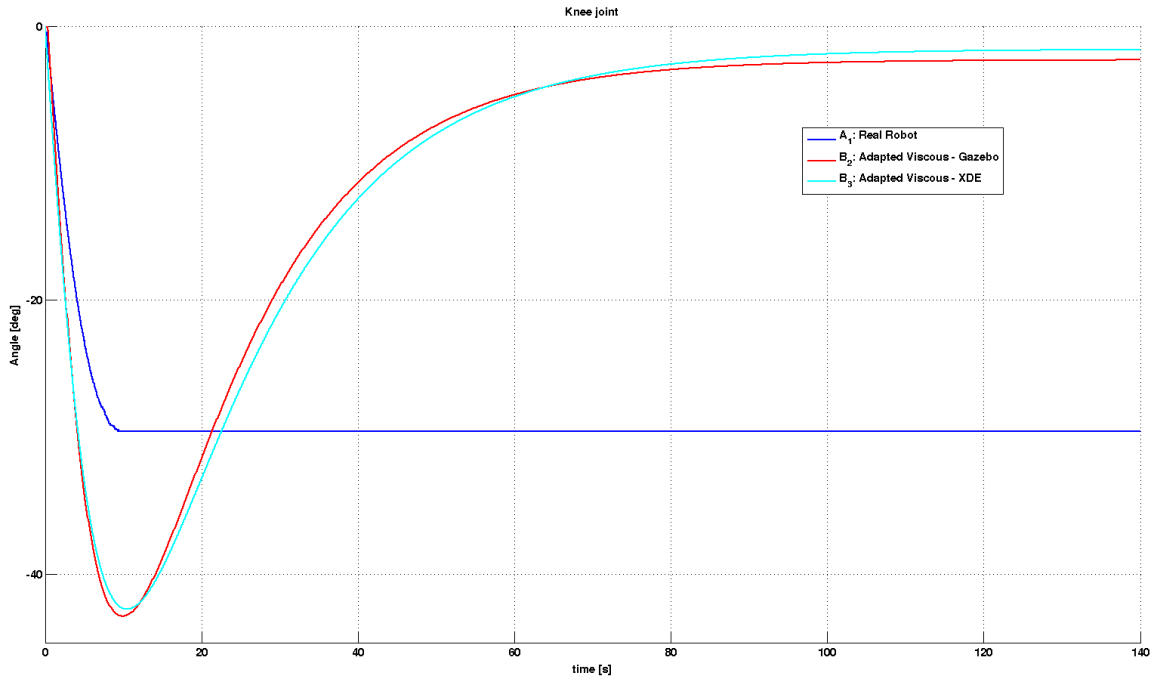


Figure 7: Evolution of the knee angle.

References

[1] R. Featherstone and D. E. Orin, *Handbook of Robotics*. B. Siciliano and O. Khatib Eds., Springer, 2008, ch. Dynamics, pp. 35–65.

[2] A. Jain, *Robot and Multibody dynamics: analysis and algorithms*. Springer, 2011.

[3] E. Todorov, "Analytically-invertible dynamics with contacts and constraints: theory and implementation in mujoco," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[4] E. Drumwright and D. Shell, "An evaluation of methods for modeling contact in multi-body simulation," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1695–1701.

[5] ——, "Extensive analysis of linear complementarity problem (lcp) solver performance on randomly generated rigid body contact problems," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 5034–5039.

[6] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[7] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1071–1076.

[8] B. Brogliato, A. ten Dam, L. . Paoli, F. Gnot, and M. Abadie, "Numerical simulation of finite dimensional multibody nonsmooth mechanical systems," *Applied Mechanics Reviews*, vol. 55, pp. 107–150, 2002.

[9] Y.-B. Jia, "Three-dimensional impact: energy-based modeling of tangential compliance," *Int. J. Robotic Research*, vol. 32, no. 1, pp. 56–83, 2013.

[10] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3982–3987.

[11] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 36–47, 2006.

[12] G. Metta, P. Fitzpatrick, and L. Natale, "Yarp: Yet another robot platform." *International Journal of Advanced Robotic Systems*, vol. 3, no. 1, 2006.

[13] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3, no. 3.2, 2009.

[14] H. Bruyninckx, P. Soetens, and B. Koninckx, "The real-time motion control core of the Orocos project," in *IEEE International Conference on Robotics and Automation*, 2003, pp. 2766–2771.

[15] E. Einhorn, T. Langner, R. Stricker, C. Martin, and H. Gross, "Mira - middleware for robotic applications," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 2591–2598.

[16] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori, "An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator," in *Proceedings of the 8th workshop on performance metrics for intelligent systems*. ACM, 2008, pp. 57–61.

[17] "The Robotcub project." [Online]. Available: http://www.robotcub.org

[18] "ODE – Open Dynamics Engine." [Online]. Available: http://ode-wiki.org

[19] "SDL – Simple Directmedia Layer." [Online]. Available: http://www.libsdl.org

[20] "XDE." [Online]. Available: http://www.kalisteo.fr/lsi/en/aucune/a-propos-de-xde

[21] "OpenHRP." [Online]. Available: http://www.openrtp.jp/openhrp3/en

[22] "Gazebo." [Online]. Available: http://gazebosim.org

[23] "Bullet Physics." [Online]. Available: http://bulletphysics.org

[24] R. Featherstone, *Rigid Body Dynamics Algorithms*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2007.

[25] "Siconos – MLCP." [Online]. Available: http://siconos.gforge.inria.fr/Numerics/MLCProblem.html

[26] S. Barthélémy, J. Salini, and A. Micaelli, "Arboris-Python." [Online]. Available: https://github.com/salini/arboris-python

[27] J. Salini, "Dynamic control for the task/posture coordination of humanoids: toward synthesis of complex activities," Ph.D. dissertation, Universit Pierre et Marie Curie, Paris, France, June 2012.

[28] "URDF - Unified Robot Description Format." [Online]. Available: http://wiki.ros.org/urdf

A Robotics simulators survey paper

Tools for dynamics simulation of robots: a survey based on user feedback

Serena Ivaldi[†,‡], Vincent Padois[†,‡] and Francesco Nori[§] [*†§]

February 27, 2014

Abstract

The number of tools for dynamics simulation has grown in the last years. It is necessary for the robotics community to have elements to ponder which of the available tools is the best for their research. As a complement to an objective and quantitative comparison, difficult to obtain since not all the tools are open-source, an element of evaluation is user feedback. With this goal in mind, we created an online survey about the use of dynamical simulation in robotics. This paper reports the analysis of the participants' answers and a descriptive information fiche for the most relevant tools. We believe this report will be helpful for roboticists to choose the best simulation tool for their researches.

1 Introduction

With the progress of powerful computers enabling fast computations, dynamics simulation in robotics is no longer expected to be an offline computational tool. It is used to rapidly prototype controllers, evaluate robots design, simulate virtual sensors, provide reduced model for model predictive controllers, supply with an architecture for real robot control, and so on.

There is a growing number of tools for dynamics simulation, ranging from dynamic solver libraries to systems simulation software, provided through either open or closed source code solutions, each more or less tailored to their expected domains of application.

The spectrum of robotics applications being large and in expansion, it is necessary for the developer community to have a feedback about the users' needs, and for the researchers to be aware of the available tools and have the elements to ponder which of the available tools is the best for their research.

With this goal in mind, we created an online survey about the use of dynamical simulation in robotics.[1] The survey was divided into four parts: general information about the user, user experience with dynamics simulation in general, user experience with one tool of his choice, technical questions and subjective evaluation about the selected tool. The survey was advertised on the main robotics mailing lists (e.g., euron-dist, robotics-worldwide) as well as in other mailing lists of correlated disciplines (e.g. comp-neuro), and kept open for approximately one month.

This paper summarizes the analysis of the users' answers. We also report a descriptive fiche for the most relevant software tools, for the reader's interest. For the complete analysis of the simulators survey, we refer the user to the extended version of the report.[2]

[*]E-mail: serena.ivaldi@isir.upmc.fr

[††] Sorbonne Universités, UPMC Paris 06, UMR 7222, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005, Paris, France.

[‡‡] CNRS, UMR 7222, Institut des Systèmes Intelligents et de Robotique (ISIR), F-75005, Paris, France.

[§§] Robotics, Brain and Cognitive Sciences Dept., Italian Institute of Technology.

[1]Online survey: `http://goo.gl/Tmyf5A`

[2]`http://www.codyco.eu/survey-simulation`

1.1 Why user feedback?

Most middleware for robotics (ROS, YARP, OROCOS, Player, etc.) are already open-source, some also cross-platforms. This makes it possible to produce interesting performance comparisons that can help the roboticists to pick the best middleware for their needs [1]. Similar ideas (open-source and cross-platform compatibility) should be used to compare dynamics models and simulators. For example, an interesting evaluation and performance comparison of contact modeling algorithms was presented in [2, 3].

As a complement to quantitative comparisons, a useful element of evaluation (often un-mentioned and neglected) is user feedback. What do users really think of the software they use for simulation? Would they suggest it? What is their experience in their particular use case? We believe user feedback may be useful to avoid time-consuming tuning and inappropriate choices of software to researchers. It could point a researcher to a community that is actively using the tool and that is sharing the same concern: for example, it is likely that people simulating flying robots have different needs than those simulating wheeled robots or those controlling bipeds. Furthermore, user feedback can provide useful suggestions to the developers community about the things that matter the most to users in simulation.

1.2 Challenges in simulation

Dynamics simulators for robotics have more strict requirements than the ones used for animating virtual characters, where time, computational burden and physical reality can be less constraining. In entertainment (e.g. video-games), unfeasible forces may not be a problem since the law of physics can be violated. In bio/mechanical studies, simulators can be used offline to analyze or synthesize behaviors. Although the field of dynamics modeling and simulation has matured over the last decades [4, 5, 6], the growing need to control whole-body movements of complex structures, such as humanoids, poses additional challenges to simulators for robotics:

1) numerical stability, which poses strong limitations on the use of simulations in real-time control settings [2, 3];

2) the capability to be used as predictive engines in real-time control loops [7], which requires the ability to be extremely fast in computing the dynamics and the guarantee for the solvers to converge to physically feasible solutions upon a certain time [8];

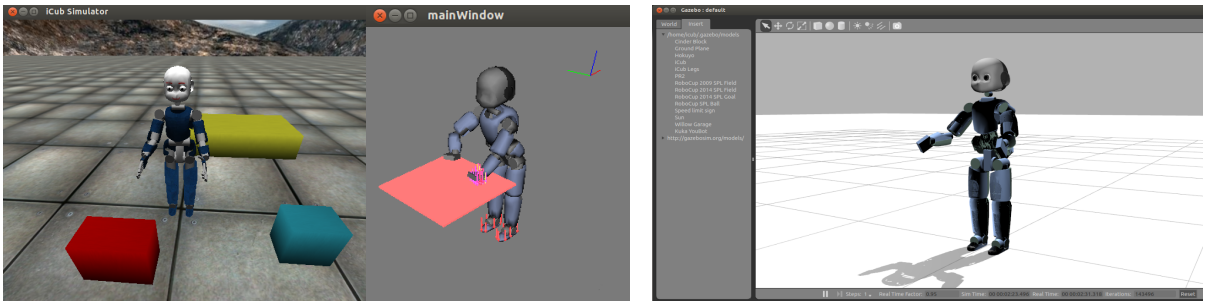
3) the simulation of rigid and soft bodies in contact with rigid and compliant environments [9, 10]: the inaccurate computation of contact forces between bodies may result in unrealistic contacts or physically unfeasible contact forces (this issue has been particularly evident in the virtual phase of the Darpa Robotics Challenge - DRC);

4) the capability to model and simulate new types of actuation systems, such as variable impedance or soft actuators [11], and different types of contacts, for example with deformable materials, compliant and soft surfaces [12].

Finally, the robotics community urges for standardized software tools and particularly open source software. The benefit of open-source is not only in the community that can grow around the software, developing new tools, improving its quality and avoiding to "re-invent the wheel" at each time, but also in checking its efficiency and robustness on real platforms (which is expensive).

1.3 The iCub case

The iCub community recently faced the problem of choosing the correct tool for whole-body dynamics simulation. The existing simulator iCubSim [13], is based on ODE and is mostly used as a tool for testing behaviors before trying them on the real robot. It is provided with an interface that emulates the low-level control of iCub, so the same code can be used to control simulated and real robot. However, the dynamics engine makes it inadequate for research about control of contacts and compliant surfaces. At the moment two solutions are investigated: one based on XDE and the other based on Gazebo. The choice of these tools has been based on objective criteria (license, developing community, stability of the software simulation), previous experience and "subjective feedback" acquired orally discussing with colleagues, that provided



(a) iCubSim, based on ODE. (b) iCub in XDE. (c) iCub simulated in Gazebo.

Figure 1: Simulators of iCub. From left to right: iCubSim, based on ODE, XDE and Gazebo. (credits for Gazebo: Silvio Traversaro)

partial and unstructured information. A more structured information about user feedback would have been helpful. We believe this survey analysis could be a further element for choosing the best simulation tool in a research project.

1.4 Comparing simulators

It is certainly difficult to enumerate all the criteria that one can examine to choose a dynamics simulator, especially for a humanoid robot that is supposed to have physical interactions with rigid and compliant environments.

First, one can choose between physics engines (e.g. ODE, Bullet) and more complex softwares that include system simulation (e.g. Gazebo, V-Rep).

Second, facing the decision to adopt a simulator for a robot, a researcher should first decide between softwares that also include system simulation, and softwares which only simulate the dynamics of multi-body systems. This criterion allows us to consider under different perspectives two set of softwares: the first set, composed of software like Gazebo, OpenHRP, iCubSIM, which facilitate seamless simulation and control of the virtual characters and their corresponding physical system/robot; the second, like Humans, OpenSIM, Robotran, that are able to simulate the dynamics of complex systems but are not meant to provide seamless control of robotics platforms.

Another element of discrimination is the way the simulator represents rigid-body structures: on one hand we have software based on ODE and Bullet, such as Gazebo, iCubSim, MORSE, which represents joints as constraints between bodies; on the other we have softwares like XDE, OpenHRP, which make use of parameterized rigid-body dynamics representations, where joints are simply part of the robotics structure. These two classes determine not only the way forward/inverse dynamics are computed (and of course the second group also benefits from the straightforward computation of quantities useful in robotics, such as Jacobians, mass matrices etc.), but most importantly the way contact forces are computed. The first class considers contacts forces as bilateral/unilateral constraints, which are added to the list of constraints used to describe the joints; then the same solver is used to find the forces for the global system, including contacts and joints. In the second class, on the contrary, only constraints from the contacts are solved, which notably simplifies the problem. In generla, finding the correct contact forces can be burdensome. Current approaches to solve this problem are mostly based on the Linear Complementarity Problem (LCP) [3], and in some cases there are mixed approaches combining LCP with optimization techniques, such as in MuJoCo [7].

In short, there are several “objective” criteria that one can look at, on the basis essentially of what is advertised by the developers as a “supported feature”. However, it is very difficult to find practical comparison of different simulators on test problems, for many reasons: first, an extensive comparison would require access to the source code but not all software is released under open-source; second, even open-source

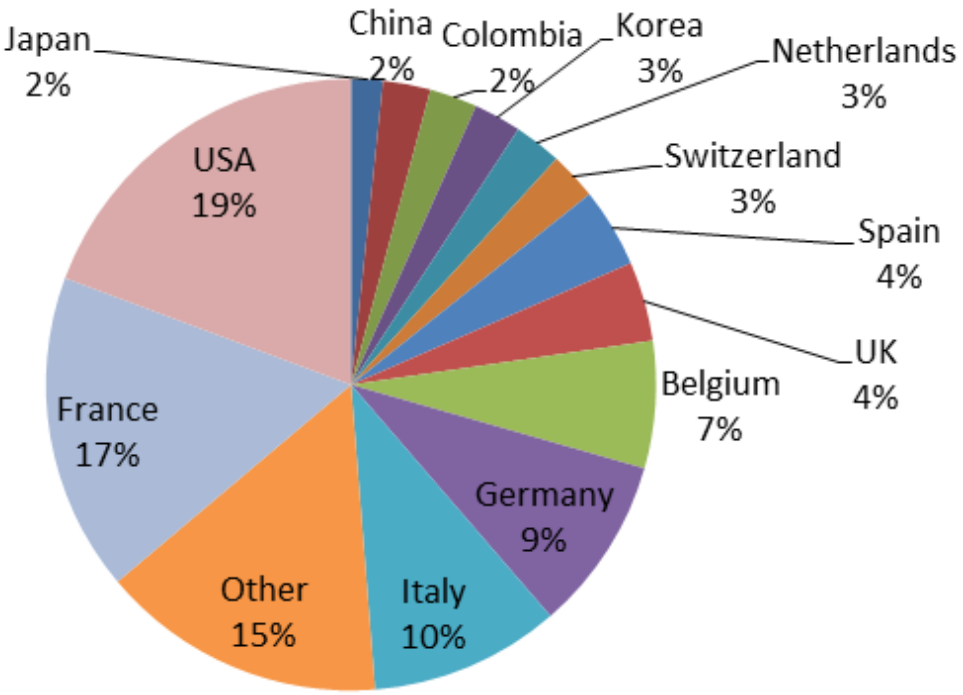


Figure 2: Country of provenience for the participants to the survey.

softwares can be difficult to compare, because their requirements in terms of architecture, dependencies etc. are different; finally, not all softwares are well-documented and easy to test in the same way, so non-experienced users may not know all the tweaks to boost simulations. We compensate the lack of objective experimental comparison with the user feedback provided by this survey.

2 Survey overview

The analysis of the survey is reported hereinafter.

2.1 About the participants

The survey was filled by 119 participants (92% male, 8% female; age 32 $\pm$ 6, min 20, max 57), whose 62% holds a PhD degree and 35% a BS or MS degree, mostly from USA, France, Italy and Germany (see Figure 2). Participants work mostly in University (70%) or do R&D in public (16%) or private (14%) institutes. Their primary areas of research are: 21% control, 14% locomotion, 10% machine learning, 9% HRI, 8% planning, 6% mechanical design, 5% cognitive robotics, 5% mathematical modeling. Their primary application field is: 26% humanoid robotics, 20% mobile robotics, 11% multi-legged robotics, 8% service robotics, 7% industrial robotics, 7% numerical simulation of physical systems, 5% flying robots. Among the participants working in humanoid robotics, 16% is also competing in the Darpa Robotics Challenge (DRC), which makes 8% of the participants to the survey - 10 people.[3]

[3]Interestingly, the software tool they indicated as the one currently used for their research (we can presume for the DRC as well) is Gazebo (3), MuJoCo (2), Robotran (2), Drake (1), Autolev (1) and ODE (1).

Tool	Currently used, and its the main tool	Currently used, but not the main tool	Currently used, just to test it	Used once, just to test it	Used then abandoned	Known, but never used	Never heard of
Gazebo	**13%**	7%	3%	18%	10%	34%	**15%**
ODE	**11%**	12%	5%	18%	**22%**	22%	**10%**
Bullet	5%	13%	7%	12%	10%	29%	24%
V-Rep	5%	3%	3%	18%	3%	29%	39%
Webots	4%	7%	1%	16%	**13%**	32%	27%
OpenRave	5%	3%	2%	7%	5%	29%	49%
Robotran	4%	0%	1%	4%	2%	13%	76%
XDE	5%	3%	0%	3%	1%	14%	74%
Blender	5%	17%	7%	22%	6%	28%	**15%**
MuJoCo	2%	0%	0%	4%	2%	21%	71%
iCub_SIM	4%	4%	2%	3%	3%	29%	55%
Nvidia PhysX	1%	1%	4%	12%	7%	43%	32%
OpenSIM	3%	4%	3%	8%	1%	41%	40%
HumanS	0%	0%	0%	1%	1%	10%	**88%**
Moby	2%	1%	0%	0%	2%	14%	81%
Vortex	3%	2%	0%	5%	5%	17%	68%
RoboRobo	3%	1%	0%	0%	1%	4%	**91%**

Table 1: Knowledge and past/present use of simulators.

2.2 General knowledge about simulating tools

We asked participants to indicate their familiarity with some of the most common existing simulation tools. We provided a list of existing software tools for simulations, used in different contexts. We asked the users to indicate whether the software was currently used or not for their researches, if it had been used before or if it was unknown. A summary of the percentage of answers for the most relevant tools is shown in Table 1.

The **software tools that have more than 5% of user share** (i.e., positive answers to the fact that the software is currently used and it is the one or one of many main tools): the most used are Gazebo (15%) and ODE (11%), with a gap with respect to Bullet, OpenRave, V-Rep, XDE and Blender, all at 5%. These values provide an indicative dimension of the user community around each software tool.

The software tools that are less known (because maybe they were not sufficiently advertised or do not have a big community behind) and the ones that are most known (even if this does not necessarily means that they are used) can be retrieved from the column "Never heard of this software" from Table 1[4]. The **most known tools** are ODE (10%), Gazebo (15%), Blender (15%), Bullet (24%), Webots (27%), Nvidia PhysX (32%), Stage (38%), V-Rep (39%), OpenSIM (40%) and ADAMS (45%). Interestingly, the first three are also open-source projects.

An important information that we acquired through the survey is about the abandon of software for simulation: this can be found in the column "Used than abandoned" in Table 1. The **most abandoned software after use** are ODE (22%), Stage (16%), Webots (13%), Bullet (10%), Gazebo (10%), Nvidia PhysX (7%), OpenHRP (6%), Blender (6%), OpenRave (5%), Vortex (5%). Though this set may seem as a sort of "blacklist" of tools that disappointed users, it must be observed that most of them are open-source softwares that could have been the "one among many" tools that have been used then in one researcher's life; however, it can be equally presumed that the high percentage of abandon can be partly correlated to the difficulty that users have encountered in using these tools and partly by their "seniority".

[4]Actually, Table 1 is only showing values for the most relevant software tools. To see the full data, we refer the reader to the full report of the survey.

Rank	Feature	Evaluation
1	Stability of simulation	Very important
2	Speed	Important
3	Precision of simulation	Important
4	Accuracy of contact resolution	Important
5	Same interface between real & simulated system	Important
6	Computational load (CPU)	Neutral
7	Computational load (memory)	Neutral
8	Visual rendering	Neutral

Table 2: Most important features for a simulator.

2.3 Important features for simulation

We asked participants to indicate the main purposes for the use of dynamics simulation in their research (they could indicate more than one): 66% simulating the interaction of the robot with the environment, 60% simulating the robot locomotion, 59% simulating behaviors of the robot before doing them on the real robot, 49% simulating the robot navigation in the environment, 48% simulating collisions and interactions between bodies (not specifically robots), 41% testing low-level controllers for robots, 22% simulating multi-fingered grasp, 21% simulating human movements, 8% animating virtual characters.

We also asked participants to evaluate, upon their experience, what are the most important features for a good simulation (they could evaluate the importance of each element from "not important at all" - 1 to "very important, crucial" - 5). Their ranking of important features is reported in Table 2. The stability of simulation is the only element that was evaluated as "very important", whereas speed, precision and accuracy of contact resolution were marked important. Remarkably, the same API between real and simulated robot is also signed as important.

2.4 Criteria for choosing a simulator

We asked participants to indicate the most important criteria for choosing a simulator. The answer was broken in three parts, i.e. participants could point out the first, second, and third most important criteria. The first most important criteria: 32% simulation very close to reality, 24% open-source, 19% same code for real and simulated robot, 11% light and fast, 6% customization, 3% no inter-penetration between bodies, 5% other. The second and third choice for the important criteria follow more or less accordingly. Considering the three criteria as a whole, i.e. grouping the three of them on the same level, the important criteria is 23% simulation very close to reality, 20% open-source, 18% light and fast, 16% same code for real and simulated robot, 14% customization, 4% no-inter-penetration between bodies, 1% ease to learn/use, 1% real time - based simulation, 2% other. If instead we consider the weight of each selection (most important=3, second important=2, third most important=1), then grouping the answers we have: 26% simulation very close to reality, 22% open-source, 17% same code for both real and simulated robot, 17% light and fast, 11% customization, 4% no inter-penetration between bodies (5% other)

2.5 Currently used tools

We asked participants to indicate the current simulation tool they are using. Results are shown in Figure 3. The most diffused software among the participants are: 13% Gazebo, 9% ARGoS, 8% ODE, 7% Bullet, 6% V-Rep, 6% Webots, 5% OpenRave, 4% Robotran, 4% XDE. All the other tools (see Figure 4) have less than 4% of user share. These tools are the ones we are focusing on in our following analysis.

Some technical information about the selected tools can be indicative of the user needs and use:

Rank	Most important criteria
1	Simulation very close to reality
2	Open-source
3	Same code for both real and simulated robot
4	Light and fast
5	Customization
6	No interpenetration between bodies

Table 3: Most important criteria for choosing a simulator.

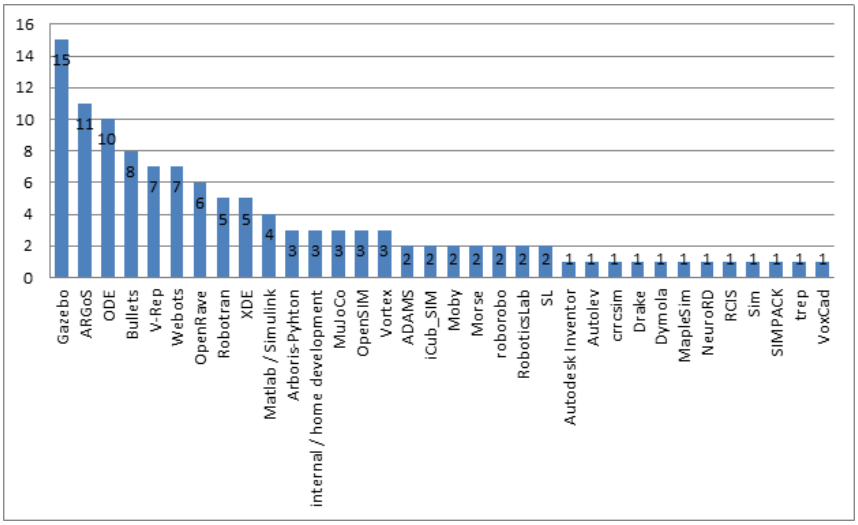


Figure 3: The simulation tools currently in use among the participants to the survey. The vertical axis reports the number of users that indicated the tool as their principal.

Research area	Users	Most used software	Other used software
Humanoid Robotics	32	(4) ODE, (3) Gazebo, Robotran, OpenRave, Arboris-Python, (2) XDE, iCub_SIM	(1) Drake, MapleSim, MuJoCo, OpenSIM, RoboticsLab, SL, Vortex, V-Rep, Webots, own code
Mobile Robotics	25	(5) Gazebo, ARGoS, (3) Webots, (2) V-Rep, Vortex	(1) ADAMS, Autodesk Inventor, Bullet, ODE, Morse, roborobo, Sim, own code
Multi-legged robotics	13	(3) Webots, (2) ODE	(1) Gazebo, ADAMS, Autolev, Bullet, Moby, RoboticsLab, SIMPACK, VoxCad
Service robotics	12	(4) Gazebo, (3) OpenRave	(1) OpenSIM, V-Rep, Morse, RCIS, SL
Numerical simulation of physical systems	8	(2) Bullet	(1) MuJoCo, ODE, OpenSIM, Simulink, trep, XDE
Flying robots	6	(2) ARGoS	(1) Robotran, crrcsim, Gazebo, Simulink/Matlab
Swarm robotics	5	(4) ARGoS	(1) roborobo
Industrial manipulators	5		(1) Bullets, Dymola, Matlab, V-Rep, XDE
Mechanical design	4		(1) Moby, MuJoCo, V-Rep, own code
Human Motion analysis	3		(1) Robotran, Bullet, XDE
Snake robots	3	(2) ODE	(1) Matlab

Table 4: Most diffused tools for a selection of the research areas.

- **Primary OS**: 66% GNU/Linux, 30% Windows, 4% MAC OSX.
- **Primary API language**: 52% C++, 18% python, 13% Matlab, 8%C, 3% LUA, 2% Java; 3% of participants do not use an API
- **License**: 67% of the tools are open-source (GPL, Apache, BSD and analogous/derivatives licenses), only 17% of the tools have a commercial license, 16% have an academic license (i.e., they are free but not open-source).
- **Hardware**: 39% a powerful desktop (i.e., multi-core, 8/16GB RAM), 35% everyday laptop, 18% powerful desktop with powerful GPU card, 5% multi-core cluster.
- **Middleware**: 52% is not using the tool with a middleware, the remainder is using ROS (25%), YARP (6%), OROCOS (4%).

The research areas being different, we extracted the most used tools for a selection of research areas: results are shown in Table 4. The most relevant results are for humanoid robotics (31 users, that is 26% of the participants to the survey) and mobile robotics (25 users, that is 21% if the participants). For humanoid robotics, the most diffused tools are ODE and Gazebo, and there is a variety of several custom-made simulators. It is interesting to notice that Gazebo supports ODE and Bullet as physical engines, hence it is probable that the quota of ODE for humanoid robotics is higher. For mobile robotics, the most diffused tools among the survey participants are Gazebo, ARGoS and Webots.

The different concentration of tools for the different research areas reveals that some tools are more appropriate than others for simulating robotic systems in different contexts or applications. A researcher may therefore let his choice about the adoption of a simulator be guided by the custom in his field. With this in mind, we investigated what was the main reason for a researcher to pick up his current tool. Overall, the main reasons why they chose the current tool is: 29% the best tool for their research upon evaluation,

Tool	Documentation	Support	Installation	Tutorials	Advanced use	Active project & community	API	Global
Gazebo	3.47±0.99	4.00±1.07	3.93±1.03	3.53±1.12	3.80±0.86	4.73±0.45	3.67±0.82	3.88±0.91
ARGoS	3.40±0.70	3.90±0.99	4.70±0.48	4.20±0.63	4.60±0.70	4.10±0.74	4.30±0.67	4.17±0.70
ODE	3.80±0.63	3.40±1.07	4.10±1.28	3.20±1.13	3.90±1.37	3.30±1.25	3.40±1.26	3.59±1.15
Bullets	3.37±1.06	3.62±0.91	4.75±0.46	4.00±0.76	3.75±0.71	4.37±0.74	3.87±0.83	3.96±0.78
V-Rep	4.28±0.76	4.43±0.79	4.71±0.76	4.14±0.90	4.28±0.76	4.43±0.53	4.14±1.07	4.25±0.80
Webots	3.86±1.07	3.57±1.13	4.43±0.79	3.43±1.51	4.42±0.78	4.14±0.69	4.57±0.53	4.20±0.96
OpenRave	3.50±0.55	4.67±0.52	4.17±0.75	3.50±1.22	4.33±0.82	4.33±0.52	4.33±0.52	4.12±0.70
Robotran	3.60±0.55	3.80±0.45	3.80±0.45	3.20±0.84	4.20±0.84	3.20±0.84	3.80±0.45	3.66±0.63
Vortex	3.33±1.15	3.67±1.53	5.00±0.00	2.67±0.58	3.67±0.58	2.67±1.15	3.33±0.58	3.48±0.80
OpenSIM	4.33±0.58	4.67±0.58	3.67±0.58	3.00±1.00	4.00±0.00	4.67±0.58	3.67±0.58	4.00±0.55
MuJoCo	2.33±1.15	1.67±0.58	4.33±1.15	3.33±1.15	4.67±0.57	4.00±0.00	5.00±0.00	3.62±0.66
XDE	1.40±0.55	2.80±1.09	3.60±0.55	2.80±1.09	3.40±1.10	2.80±0.84	3.00±1.00	2.83±1.07

Table 5: Ratings for the level of user satisfaction of the most diffused tools.

23% "inheritance", i.e. it was "the software" (already) used in their laboratory, 8% they are the developers, 8% it was chosen by their boss/project leader, 7% it is open-source, 7% it was happily used by colleagues. Only 3% of the participants chose the tool because of a robotic challenge. Interestingly there is quite a demarcation between the first reasons and the others. There are certainly some tools that distinguish for the fact that they have been chosen as best option for research, for example V-Rep (71%), Bullet (63%) and Gazebo (53%). Some tools have instead been adopted by "inheritance", i.e., they were already used in the lab: ARGoS (45%), Robotran (40%) and XDE (40%). For the latter, it is also a choice imposed by the project leader (40%).

We asked participants to evaluate their level of satisfaction of the use of their tool, in a global way, from Very negative (1) to Very Positive (5): all software tools were evaluated "positive", whereas only MuJoCo was "very positive" (subjective evaluation by 3 users). We also asked participants to indicate their level of satisfaction with respect to some specific aspects (documentation, support, installation, tutorials, advanced use, active project and community, API), and to rate each element on a scale from 1 to 5. Table 5 reports the mean and standard deviation of the notes received by the users of each tool.

2.6 Tools for robots

The majority of participants to the survey is using the software tool to simulate robots (91%). We extracted the principal tools used for simulating the main robots:

- iCub: 25% Arboris-Python, 17% ODE, 17% Robotran, 17% iCub_SIM
- Atlas: 50% Gazebo, 25% MuJoCo, 12% Autolev, 12% Drake
- PR2: 21% OpenRave, 14% Gazebo, 14% MuJoCo, 7% Bullet, 7% V-Rep
- Multi-legged robot: 22% ODE, 11% SL, 11% Bullet, 11% Webots
- Wheeled vehicle: 14% Gazebo, 14% V-Rep, 11% ARGoS, 7% Morse, 7% Webots, 7% Vortex
- Quadrotor: 24% Gazebo, 24% ARGoS, 12% V-Rep

3 Software information fiches

We report in the following some essential information for the main software tools (the most diffused) that may be of help for the interested reader. Most of the information gathered here is extracted from the survey (each item is marked by a filled dot, •). When it is not the case, an empty dot ◦ is used. For the subjective user feedback we refer the reader to the full report of the survey. Data are reported with %, however to have a fair comparison we report in brackets the number of participants that selected the specified tool. Note that in the following "main simulated robots" refers to real robots that are simulated in the software.

3.1 Gazebo

Gazebo is a multi-robot simulator for outdoor environments, developed by Open-Source Robotics Foundation. It is the official software tool for the DRC. It supports multiple physics engines (ODE, Bullet).

- ◦ Web: `http://gazebosim.org/`
- ◦ License: Apache 2
- • Survey participants: 15
- • OS share: 100% GNU/Linux
- • Main API: 80% C++
- • Main reason for adoption: 53% best tool upon evaluation, 20% software already used in the lab, 20% official tool for a challenge, 7% open-source
- • Mostly used in USA (33%)
- • Mainly used for: 33% mobile robotics, 27% service robotics, 20% humanoid robotics
- • Main simulated robots: 40% Atlas, 33% custom platform, 27% wheeled vehicle, 27% quadrotor, 27% turtlebot, 20% PR2
- • Main middleware used with: 93% ROS
- • Main simulated robots: 40% Atlas, 33% custom platform, 27% wheeled vehicle, 27% quadrotor, 27% turtlebot, 20% PR2

3.2 ARGoS

ARGoS is a multi-robot, multi-engine simulator for swarm robotics, initially developed within the Swarmanoid project[5].

- ◦ Web: `http://iridia.ulb.ac.be/argos/`
- ◦ License: GPLv3.0
- • Survey participants: 11
- • OS share: 91% GNU/Linux, 9% MAC OSX
- • Main API: 73% C++
- • Main reason: 45% software already used in the lab, 27% colleagues using it
- • Mostly used in Belgium (36%) and Italy (27%)
- • Used for: 46% mobile robotics, 36% swarm robotics, 18% flying robots
- • Main simulated robots: 64% khepera/e-puck/thymio, 36% marXbot/footbot, 27% quadrotor

[5] `http://www.swarmanoid.org/`

3.3 ODE

ODE (Open Dynamics Engine) is an open-source library for simulating rigid body dynamics, used in many computer games and simulation tools. It is used as physics engines in several robotics simulators, such as Gazebo and V-Rep.

- Web: `http://www.ode.org/`
- License: GNU LGPL and BSD
- Survey participants: 10
- OS share: 100% GNU/Linux
- Main API: 80% C++
- Main reason: 50% best tool upon evaluation, 20% used before, 10% boss choice, 10% open-source, 10% software already used in the lab
- Mostly used in France (20%)
- Used for: 50% humanoid robotics, 20% multi-legged robotics, 20% snake robots, 10% numerical simulation of physical systems
- Main simulated robots: 40% multi-legged robot, 20% iCub

3.4 Bullet

Bullet is an open-source physics library, mostly used for computer graphics and animation. The latest release[6] also supports Featherstone's articulated body algorithm and a Mixed Linear Complementarity Problem solver, which makes it suitable for robotics applications.

- Web: `http://bulletphysics.org`
- License: ZLib license, free for commercial use
- Survey participants: 8
- OS share: 50% Windows, 38% GNU/Linux, 12% MAC OSX
- Main API: 75% C++
- Main reason: 63% best tool upon evaluation, 25% open-source, 12% colleagues using it
- Mostly used in France (25%), Italy (25%) and Belgium (25%)
- Used for: 25% humanoid robotics, 25% numerical simulation of physical systems, 12.5% industrial manipulators, 12.5% human motion analysis, 12.5% mobile robotics, 12.5% multi-legged robotics
- Main simulated robots: 25% multi-legged robot

[6] At the time we are submitting this paper, the latest version is 2.82, released at the end of october 2013 - after the survey.

3.5 V-Rep

V-Rep is a robot simulator software with an integrated development environment, produced by Coppelia Robotics. Like Gazebo, it supports multiple physics engines (ODE, Bullet, Vortex).

- Web: `http://www.coppeliarobotics.com/`
- License: Dual-licensed source code: commercial or GNU GPL
- Survey participants: 7
- OS share: 57% GNU/Linux, 43% Windows
- Main API: 57% C++, 29% LUA
- Middleware: 43% ROS, 57% None
- Main reason: 72% best tool upon evaluation, 14% colleagues using it, 14% boss choice
- Used for: 29% mobile robotics, 14% industrial manipulators, 14% humanoid robotics, 14% mechanical design, 14% cognitive architectures, 14% service robotics
- Main simulated robots: 29% Nao, 29% quadrotor, 29% wheeled vehicle, 29% Bioloid, 29% khepera/ e-puck/ thymio

3.6 Webots

Webots is a development environment used to model, program and simulate mobile robots developed by Cyberbotics Ltd.

- Web: `http://www.cyberbotics.com`
- License: Commercial or limited features free academic license
- Survey participants: 7
- OS share: 57% GNU/Linux, 29% Windows, 14% MAC OSX
- Main API: 71% C++
- Main reason: 29% best tool upon evaluation, 29% software already used in the lab, 14% boss choice, 14% official tool for a challenge, 14% used before
- Used for: 43% mobile robotics, 43% multi-legged robotics, 14% humanoid robotics
- Main simulated robots: 29% KUKA LWR, 29% Lego Mindstorm, 29% wheeled vehicle

3.7 OpenRave

OpenRave is an environment for simulating motion planning algorithms for robotics.

- Web: `http://openrave.org/`
- License: LGPL and Apache 2
- Survey participants: 6
- OS share: 100% GNU/Linux
- Main API: 83% python

- Main reason: 50% best tool upon evaluation, 33% colleagues using it, 17% boss choice
- Mostly used in USA (33%)
- Used for: 50% humanoid robotics, 50% service robotics
- Main simulated robots: 50% PR2

3.8 Robotran

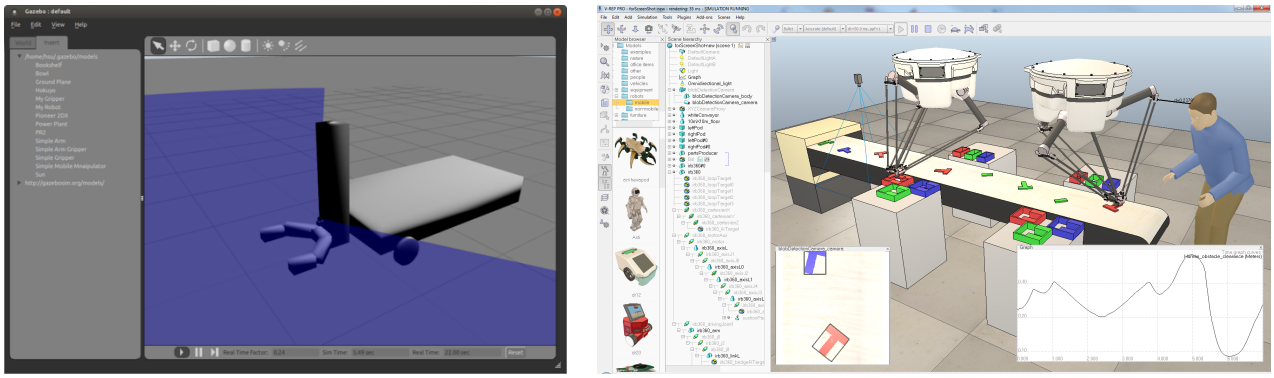
Robotran is a software that generates symbolic models of multi-body systems, which can be analysed and simulated in Matlab and Simulink. It is developed by the Center for Research in Mechatronics, Université Catholique de Louvain.

- Web: `http://www.robotran.be/`
- License: commercial and free non commercial license
- Survey participants: 5
- OS share: 80% Windows, 20% GNU/Linux
- Main API: 60% C
- Main reason: 40% software already used in the lab, 20% best tool upon evaluation, 20% developer, 20% open-source (free)
- Used only in Belgium (40%) and Italy (60%)
- Used for: 60% humanoid robotics, 20% human motion analysis, 20% flying robots
- Main simulated robots: 60% Coman, 40% iCub

3.9 XDE

XDE is an interactive physics simulation software environment fully developed by CEA LIST.

- Web: `http://www.kalisteo.fr/lsi/en/aucune/a-propos-de-xde`
- License: Commercial and free non commercial license
- Survey participants: 5
- OS share: 60% GNU/Linux, 40% Windows
- Main API: 100% python
- Middleware: OROCOS
- Main reason: 40% boss choice, 40% software already used in the lab, 20% developer
- Used only in France (100%)
- Used for: 40% humanoid robotics, 20% industrial manipulators, 20% numerical simulation of physical systems, 20% human motion analysis
- Main simulated robots: 40% industrial robots, 40% KUKA LWR, 20% iCub, 20% wheeled vehicle



(a) Gazebo. (b) V-Rep.

Figure 4: The simulation environment of Gazebo and V-Rep (credits: `http://gazebosim.org` and `http://www.coppeliarobotics.com`).

4 Conclusions

With the growing interest of robotics for physical interaction, simulation is no longer a tool for offline computation and visualization, but is used in particular for rapidly prototyping controllers. That is why researchers stressed the importance of more realistic simulation, same code for both real and simulated robot, beside the availability of the source code.

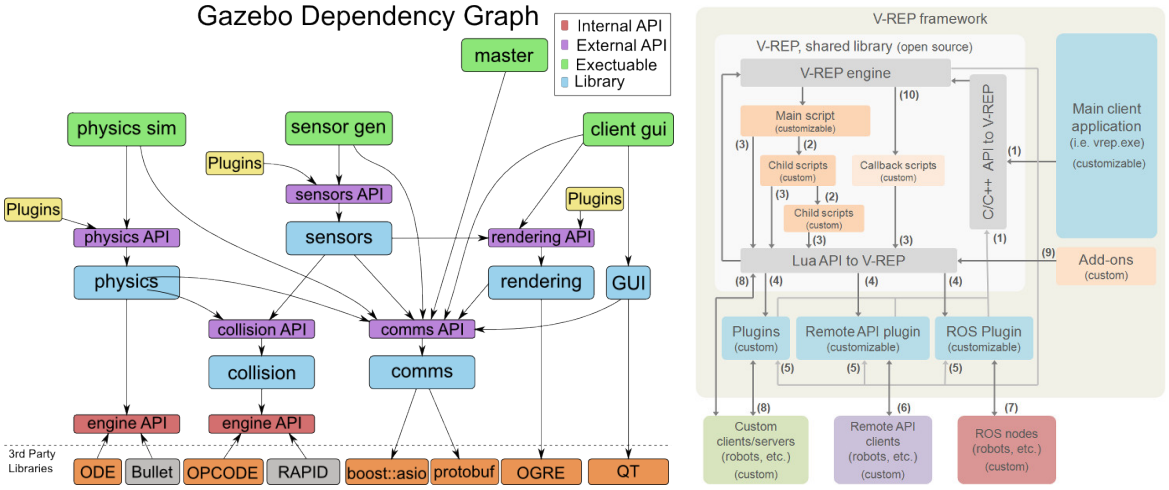
This shift in the expectations from simulation reflects in the migration from physics engines classically used for animation of virtual characters and computer graphics towards physics engines supporting robotics descriptions of bodies and more contact solvers. The users' knowledge of multiple simulation tools and their activity in testing and abandoning eventually a tool, suggest that users look for the right tool that meets their requirements and is fit for their problem. For instance, the robotics community demands physics engines with direct support of robotics descriptions of multi-body systems. This is the reason why Bullet is now supporting LCP solvers and Featherstone's ABA, and new physics engines like MuJoCo[7] or Vortex have been created.

A good compromise is a modular software that supports multiple physical engines, enabling a tradeoff between simulation accuracy and computational resources. Those features, together with the stability of the simulation, are of main concern for the users. This strategy, adopted with Gazebo by the research community and with V-Rep at industrial level, seems to pay off in terms of user feedback, because the first is the most diffused among the survey participants and the second the best rated. Subjective free-comments[8] reported that users of those tools, though acknowledging their current limitations, were confident in the announced developments that could sensibly improve the tools.

To conclude, we overviewed the panorama of simulation tools that are currently used in robotics. Each software inherits its specificities from the expected domains of application or the original application for which is was conceived, which results in a variety of tools with different features ranging from dynamic solver libraries to systems simulation software. More recent tools, like Gazebo and V-Rep, have the potential to be of general use thanks to their good support and community and the support of different physical engines. Notwithstanding, we remind that designing a perfect physics engine is impossible and there will always be a difference between simulation and reality, a gap that should be taken into account by the simulator and the robot controllers [14].

[7] MuJoCo is not merely a physics engine, it incorporates control and optimization modules.

[8] They can be read in the extended version of the survey report: `http://www.codyco.eu/survey-simulation`.



(a) Gazebo (b) V-Rep

Figure 5: A graphical representation of the software architectures of Gazebo and V-Rep (credits: `http://gazebosim.org` and `http://www.coppeliarobotics.com`).

Acknowledgment

The authors are supported by the EU Project CODYCO (FP7-ICT-2011-9, No. 600716).

References

[1] E. Einhorn, T. Langner, R. Stricker, C. Martin, and H. Gross, "Mira - middleware for robotic applications," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 2591–2598.

[2] E. Drumwright and D. Shell, "An evaluation of methods for modeling contact in multibody simulation," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1695–1701.

[3] ——, "Extensive analysis of linear complementarity problem (lcp) solver performance on randomly generated rigid body contact problems," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 5034–5039.

[4] R. Featherstone and D. E. Orin, *Handbook of Robotics*. B. Siciliano and O. Khatib Eds., Springer, 2008, ch. Dynamics, pp. 35–65.

[5] A. Jain, *Robot and Multibody dynamics: analysis and algorithms*. Springer, 2011.

[6] E. Todorov, "Analytically-invertible dynamics with contacts and constraints: theory and implementation in mujoco," in *IEEE Int. Conf. on Robotics and Automation*, 2014.

[7] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2012, pp. 5026–5033.

[8] E. Todorov, "A convex, smooth and invertible contact model for trajectory optimization," in *IEEE Int. Conf. on Robotics and Automation*, 2011, pp. 1071–1076.

[9] B. Brogliato, A. ten Dam, L. . Paoli, F. Gnot, and M. Abadie, "Numerical simulation of finite dimensional multibody nonsmooth mechanical systems," *Applied Mechanics Reviews*, vol. 55, pp. 107–150, 2002.

[10] Y.-B. Jia, "Three-dimensional impact: energy-based modeling of tangential compliance," *Int. J. Robotic Research*, vol. 32, no. 1, pp. 56–83, 2013.

[11] C. Duriez, "Control of elastic soft robots based on real-time finite element method," in *IEEE International Conference on Robotics and Automation*. IEEE, 2013, pp. 3982–3987.

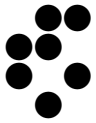
[12] C. Duriez, F. Dubois, A. Kheddar, and C. Andriot, "Realistic haptic rendering of interacting deformable objects in virtual environments," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 1, pp. 36–47, 2006.

[13] V. Tikhanoff, A. Cangelosi, P. Fitzpatrick, G. Metta, L. Natale, and F. Nori, "An open-source simulator for cognitive robotics research: the prototype of the icub humanoid robot simulator," in *8th Workshop on Performance Metrics for Intelligent Systems*, 2008, pp. 57–61.

[14] J.-B. Mouret, S. Koos, and S. Doncieux, "Crossing the reality gap: a short introduction to the transferability approach," in *"Evolution in Physical Systems" Workshop in ALIFE*, 2012.

B Technical paper on the digital human URDF file generator

“Jožef Stefan” Institute Ljubljana, Slovenija



IJS Technical report xxxx

BodyModel 1.0
Quick User Manual

Leon Žlajpah, Jan Babič

Ljubljana, 2014

Contents

1 Introduction

For simulation of human body using standard simulation environments, the model a human body is prepared using standard descriptions used for robot systems. One of them is the Unified Robot Description Format (URDF) [1], which is an XML format for representing a robot model. Although the URDF specification is intended for serial link robot manipulators, it can be also used to describe a human body. Another XML based specification is the MuJoCo XML model format [3]. A model is built by different elements (links, joints, visual representation objects, ...), which have different attributes (position, orientation, child, ...). For details on model building check the specifications for each model type [1, 3].

The **BodyModel** is used to update values of some model parameters of a human body in these models. The parameters which can be updated are:

- joint position
- link length
- link mass and center of mass (COM)
- link inertia
- some additional dimensional parameters

The values of these parameters depend on two parameters:

- Body height
- Body weight

and are calculated according to the data published by Zatsiorsky [4] and Leva [2]. Note that all units are SI (m, kg), except the input of human body is for convenience in cm.

BodyModel allows to use for model attributes in the model description some predefined variables instead of constant values. These variables are then replaced by **BodyModel** with values according to the body height and weight. When preparing the model, there is no limitation in the structure of the model, i.e. links and joints of a human body (or part of a body) used in the model. The model structure is not changed by the **BodyModel**.

2 Human body parameters

To define the human body parameters, the body is divided into parts as shown in Fig. 1:

- Head
- Torso_U (upper part of the torso)
- Torso_M (middle part of the torso)
- Torso_L (lower part of the torso)
- UpperArm (left and right)
- LowerArm (left and right)
- Hand (left and right)
- Thigh (left and right)
- Shank (left and right)
- Foot (left and right)

and each of this parts has inertial parameters

- `Mass`
- `COM` (vertical distance of center of mass from parent joint)
- `Inertia.x`, `Inertia.y` and `Inertia.z` (moments around principal axes)

and geometric parameters

- Length
- Height
- Width

Note that only some geometric parameters are defined. These parameters are listed in Table 1, where their value is explained using distances between points (as shown in Fig. 1: the points $\mathbf{J}_i$ denote joint i position, C_i denote COM of corresponding body, and P_i are some auxiliary points). For more explanation see Fig. 2 and [2].

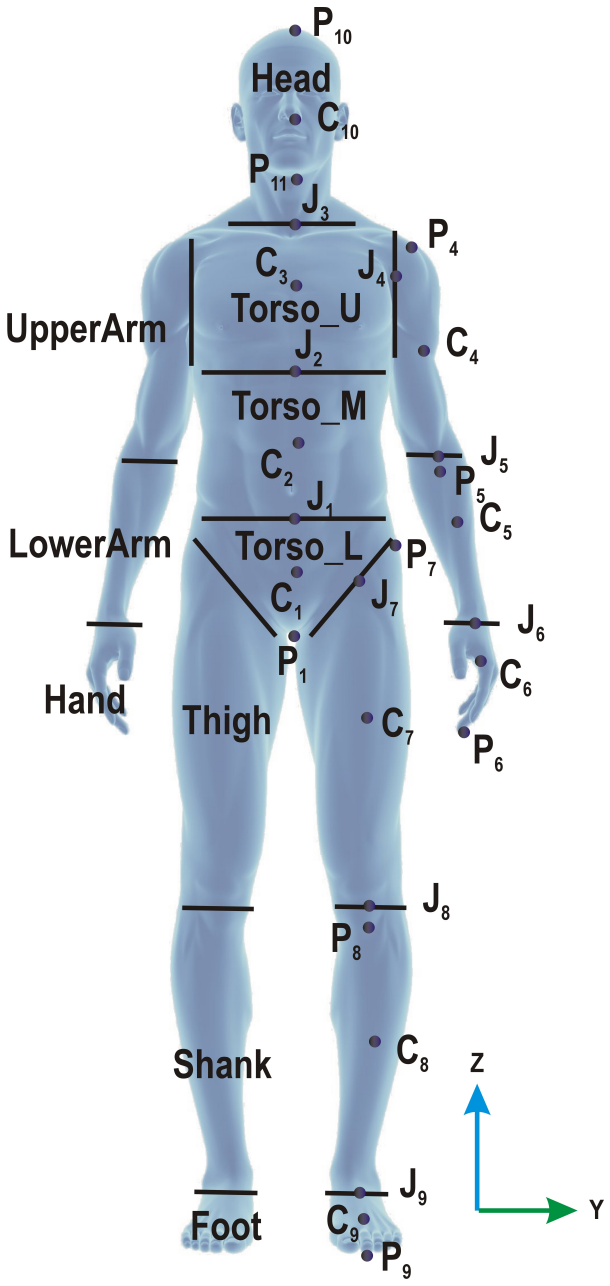


Figure 1: Body parts, joint locations (J), COM (C), aux. points (P)

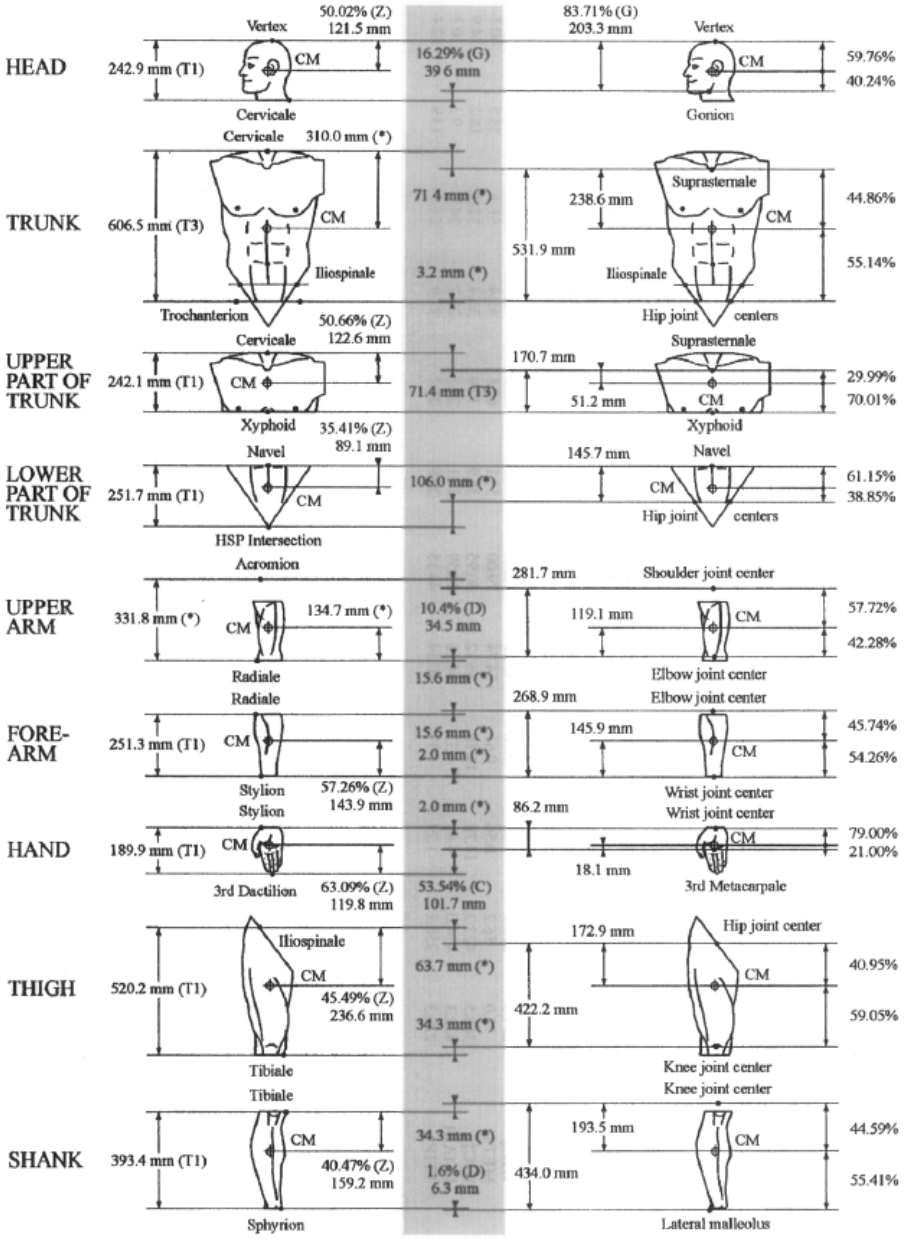


Fig. 1. A graphic description of the main adjustments to the relative CM positions for males. The adjusted distances are shown on the right of the shaded area. For all segments, except trunk and upper arm (see text), the shaded area indicates the longitudinal distances between original (on its left) and new (on its right) reference points. All percent values are relative to the segment lengths indicated on their left. (* = see text; C = Clauser *et al.*, 1969; D = de Leva, 1996; G = Gordon *et al.*, 1989; T1 = Table 1; T3 = Table 3; Z = Zatsiorsky *et al.*, 1990a).

Figure 2: Fig. 1 from [2]

Table 1: Dimensional model parameters: Height and Length are in vertical direction (z); Width is horizontal dimension (y); COM is in longitudinal direction (z, except foot in y)

Parameter	Value
`Head.Height`	$\mathbf{P}_{10} - \mathbf{J}_3$
`Torso_U.Height`	$\mathbf{P}_{11} - \mathbf{J}_2$
`Torso_M.Height`	$\mathbf{J}_2 - \mathbf{J}_1$
`Torso_L.Height`	$\mathbf{J}_1 - \mathbf{P}_1$
`Thigh.Height`	$\mathbf{P}_7 - \mathbf{P}_8$
`Shank.Height`	$\mathbf{P}_8 - \mathbf{J}_9$
`Foot.Height`	$\mathbf{J}_9 - \mathbf{P}_9$
`UpperArm.Height`	$\mathbf{P}_4 - \mathbf{P}_5$
`LowerArm.Height`	$\mathbf{P}_5 - \mathbf{J}_6$
`Hand.Height`	$\mathbf{J}_6 - \mathbf{P}_6$
`Head.Length`	$\mathbf{P}_{10} - \mathbf{P}_{11}$
`Torso_U.Length`	$\mathbf{J}_3 - \mathbf{J}_2$
`Torso_M.Length`	$\mathbf{J}_2 - \mathbf{J}_1$
`Torso_L.Length`	$\mathbf{J}_1 - \mathbf{J}_7$
`Thigh.Length`	$\mathbf{J}_7 - \mathbf{J}_8$
`Shank.Length`	$\mathbf{J}_8 - \mathbf{J}_9$
`Foot.Length`	$\mathbf{J}_9 - \mathbf{P}_9$
`UpperArm.Length`	$\mathbf{J}_4 - \mathbf{J}_5$
`LowerArm.Length`	$\mathbf{J}_5 - \mathbf{J}_6$
`Hand.Length`	$\mathbf{J}_6 - \mathbf{P}_6$
`Torso_U.Width`	distance between left and right $\mathbf{J}_4$
`Torso_L.Width`	distance between left and right $\mathbf{J}_7$
`Head.COM`	$\mathbf{C}_{10} - \mathbf{J}_3$
`Torso_U.COM`	$\mathbf{C}_3 - \mathbf{J}_2$
`Torso_M.COM`	$\mathbf{J}_2 - \mathbf{C}_2$
`Torso_L.COM`	$\mathbf{J}_1 - \mathbf{C}_1$
`Thigh.COM`	$\mathbf{J}_7 - \mathbf{C}_7$
`Shank.COM`	$\mathbf{J}_8 - \mathbf{C}_8$
`Foot.COM`	$\mathbf{J}_9 - \mathbf{C}_9$
`LowerArm.COM`	$\mathbf{J}_4 - \mathbf{C}_4$
`Upperarm.COM`	$\mathbf{J}_5 - \mathbf{C}_5$
`Hand.COM`	$\mathbf{J}_6 - \mathbf{C}_6$

2.1 Updating variables

In **BodyModel** a simple calculator is implemented, which allows to use some basic mathematical operations using the predefined variables. The predefined variables have the following form:

`<Body part>.<variable>` like, e.g. `Head.Height`

or

<Body part>.<variable>.<axis> like, e.g. Head.Inertia.x

The expression has to be enclosed in braces

{<expression>} like, e.g. {Head.Height/2+0.1}

and this expression is then replaced in the model file by the value of the expression.

For example, the URDF model of head can defined as

```
<link name="head">
  <inertial>
    <origin xyz="0 0 {Head.COM}" rpy="1.57079633 0 0"/>
    <mass value="{Head.Mass}"/>
    <inertia ixx="Head.Inertia.x"  ixy="0" ixz="0"
             iyy="Head.Inertia.y" iyz="0"
             izz="Head.Inertia.z" />
  </inertial>

  <visual>
    <origin xyz="0 0 {Head.Height-Head.Length/2}"
            rpy="1.57079633 0 0" />
    <geometry>
      <sphere radius="{Head.Length/2}" />
    </geometry>
  </visual>
</link>
```

and after **BodyModel** has updated the expression the result is

```
<link name="head">
  <inertial>
    <origin xyz="0 0 0.1185" rpy="1.57079633 0 0"/>
    <mass value="5.238"/>
    <inertia ixx="0.0289"  ixy="0" ixz="0"
             iyy="0.0314" iyz="0"
             izz="0.0214" />
  </inertial>

  <visual>
    <origin xyz="0 0 0.1378"
            rpy="1.57079633 0 0" />
    <geometry>
      <sphere radius="0.0992" />
    </geometry>
  </visual>
</link>
```

3 Installation

BodyModel is written in Pascal language using Lazarus IDE and can be compiled on Windows or Linux systems. The distribution contains the following files

`BodyModel_32.exe`	Windows executable (32 bit system)
`BodyModel_64.exe`	Windows executable (64 bit system)
`BodyModel_32`	Linux executable (64 bit system)
`BodyModel_64`	Linux executable (32 bit system)
`Body_MuJoCo.xml`	Sample MuJoCo model of a human body

No installation is needed to run the application. On Windows systems run

```
BodyModel.exe
```

and on Linux systems

```
BodyModel
```

(Make the file executable if necessary using `chmod +x BodyModel_32`)

4 Basic operation

Fig. 3 shows the main **BodyModel** window. The user can

- load the model file
- save the updated model file
- select the body height and weight
- prepare the body variables
- edit the model
- edit the attributes in the XML model file
- update the expressions in the model

4.1 Load model

In **BodyModel** any text file can be loaded. However, as most model descriptions files are using XML format, the **BodyModel** provides special Tree view for XML files.

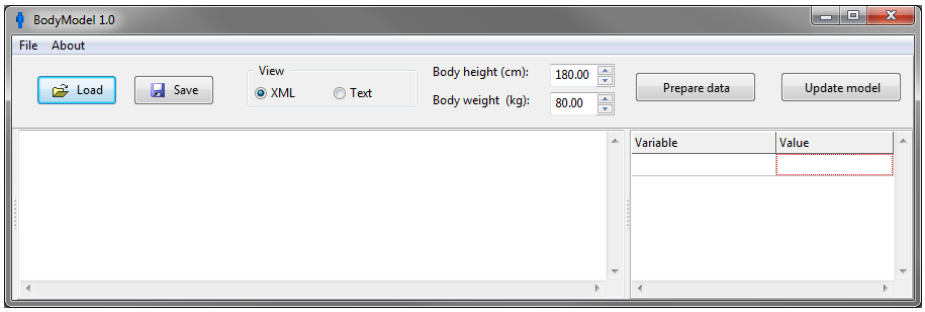


Figure 3: **BodyModel** - User interface

Figs. 4 and 5 show the standard Text view and XML tree view, respectively. The XML tree view is generated from XML file and is showing the overview of the model structure. The user can switch simply between them by selecting the corresponding radio button.

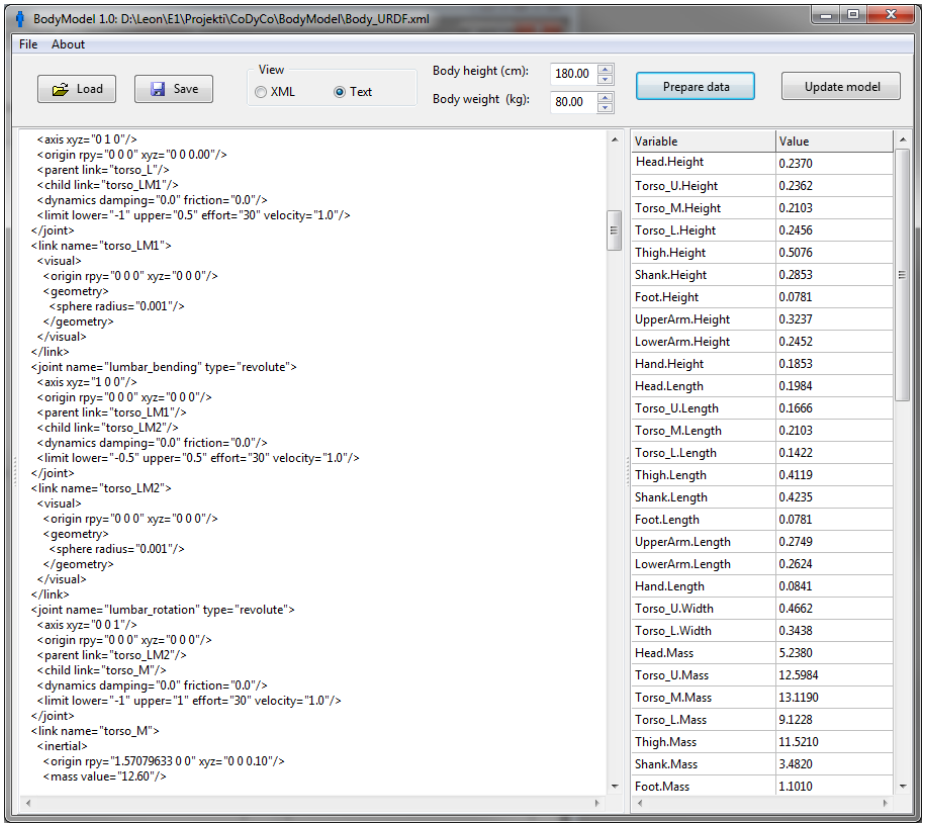


Figure 4: **BodyModel** - text view of the model

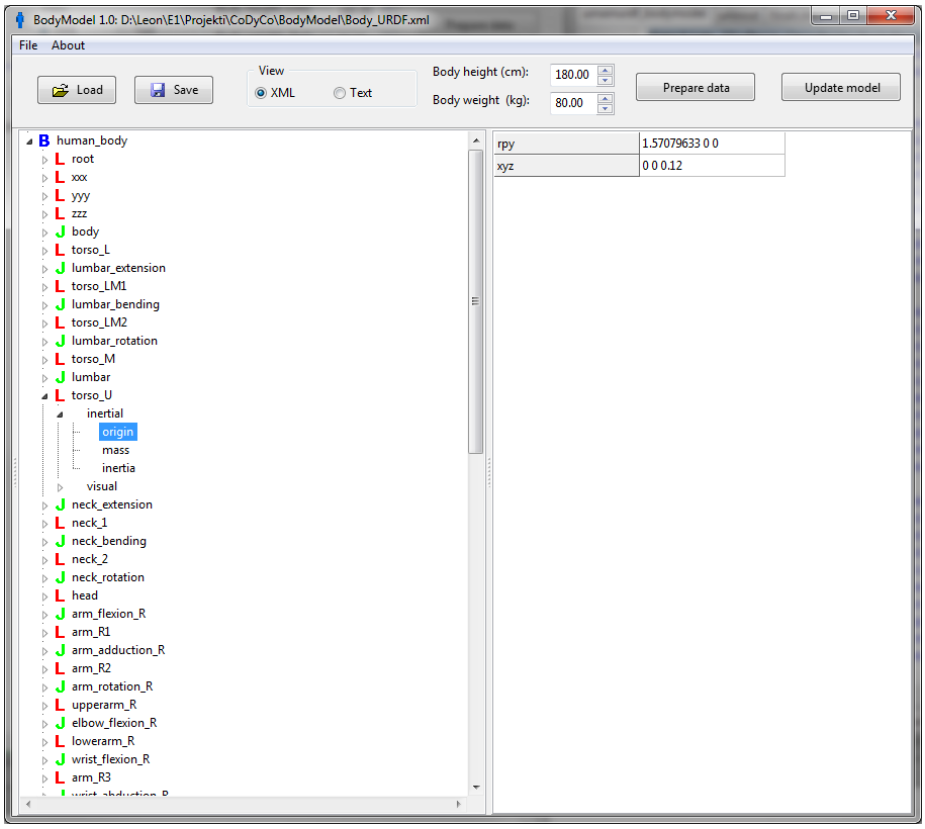


Figure 5: **BodyModel** - XML view of the model

4.2 Model editing

In normal Text view (see Fig. 4), a simple text editor enables manual changes of the model file.

In XML tree view the table on the right side shows the attributes of the selected XML element. The user can edit the attribute values.

4.3 Preparing variables

To prepare data for predefined variables, the user has to input desired body height and weight. Pressing the Update data button, the values of predefined variables are calculated and the list of all variables is shown on the right side of model file in Text view (see Figs. 4).

4.4 Model update

The model update depends on the selected view. In Text view, pressing the Update model button expressions in the model are replaced by the calculated values and if the load file is a valid XML model file, the Tree view is updated.

In XML tree view pressing the Update model button the model in the Text view is updated according to XML tree data.

4.5 Save model

After the model has been updated it can be saved bz pressing fboxSave button.

4.6 Model example

The model for the model file `Body_MuJoCo.xml`, which describes the human body for MuJoCo simulator [3], is shown in Fig. 6.

Acknowledgement

The work presented in this document was supported by the European Community Framework Programme 7 through the CoDyCo project, contract no. 600716.

References

[1] URDF Tutorial. `http://wiki.ros.org/urdf/Tutorials/Create your own urdf file`.

[2] Paolo de Leva. Adjustments to Zatsiorsky-Seluyanov's segment inertia parameters. *Journal of Biomechanics*, 29(9):1223 – 1230, 10 1996.

[3] Emo Todorov, Yuval Tassa, and Tom Erez. *MuJoCo: Modeling and simulation of Multi-Joint dynamics with Contact*, 0.5.0 edition, June 2013.

[4] V. Zatsiorsky and V. Seluyanov. *Biomechanics VIII-B*, chapter The mass and inertia characteristics of the main segments of the human body, pages 1152–1159. Human Kinetics. 1983.

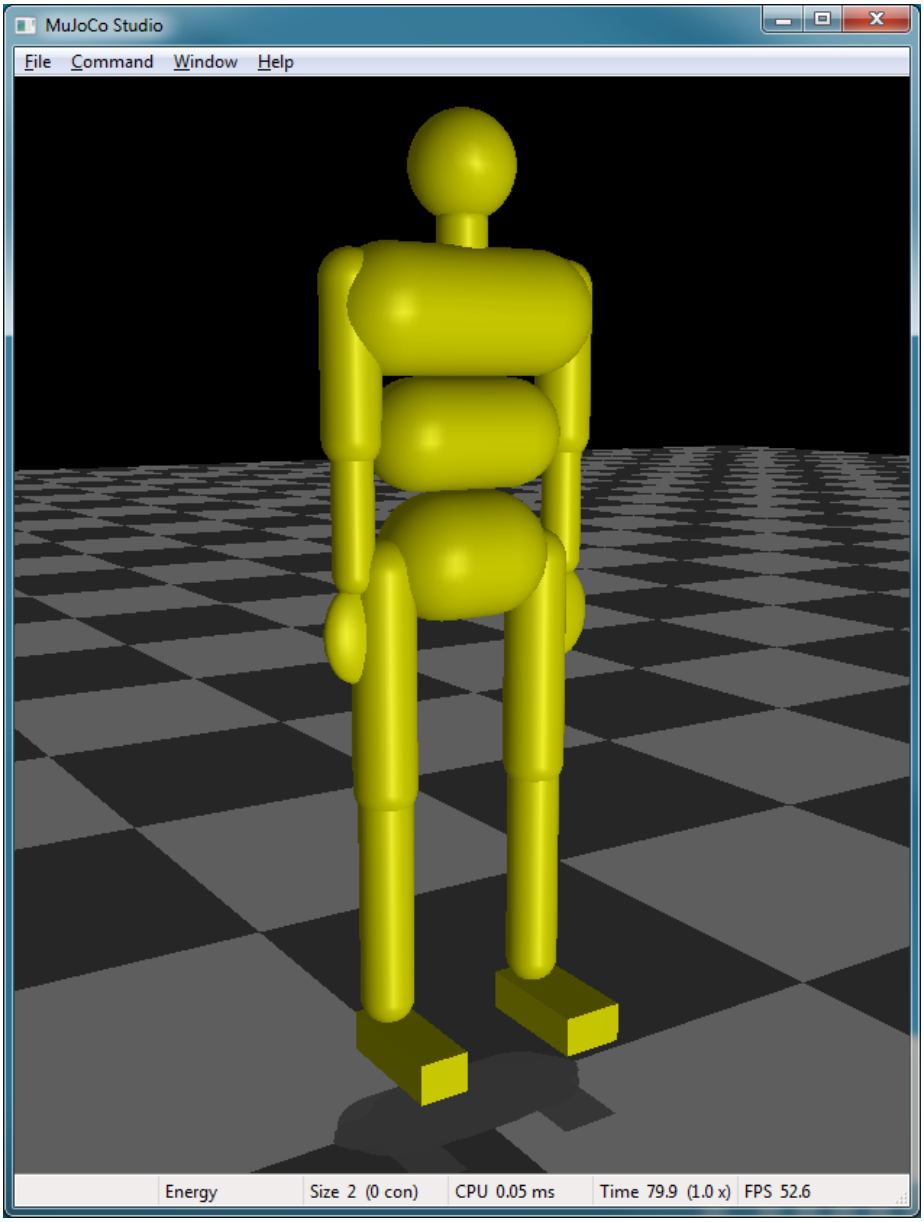


Figure 6: Model of a human body in MuJoCo simulator - model parameters are for body height 180cm and weight 80kg