OpenSim Real-Time Robotics Operating System Integration*

Frederico B. Klein[1] and Ruoli Wang[2]

Abstract— **Real-time accurate description of human movement is necessary for applications that require a close interface between robotic devices and people. The open-source simulator OpenSim is an accurate implementation of rigid body mechanics of human movement, but while written in C++, it was not designed with real-time operation as its focus. OpenSim real-time (OpenSimRT), is a step towards bridging this gap, however the absence of a framework made it harder to integrate with other systems. The work presented here aims to solve this integration issue by implementing OpenSimRT's pipeline in a Robotics Operating System (ROS) framework using Docker for easy replication. Benefits include abstraction of hardware interfaces with devices such as Inertial Motion Units (IMUs) and ease of integration with other ROS algorithms (e.g. Augmented Reality). We finally justify this approach (OpenSimRT and ROS-Docker) as a desirable platform for prototyping systems requiring accurate musculoskeletal modelling of human movement and integration with various sub-components.**

I. INTRODUCTION

Accurate human movement description can benefit many tasks such as exoskeleton control, movement analysis, human-computer interaction, movement related augmented and virtual reality design. A way to ensure accurate human movement description is by using validated movement description systems as opposed to ad-hoc solutions. However, for most of these tasks real-time estimation of movement parameters are a requirement, restricting the type of algorithms and implementations possible.

Commonly, musculoskeletal modelling and simulation can be done by optimization algorithms running a pipeline which is not meant to be executed in real-time. As such OpenSim [1], an open-source software that allows creation of models of musculoskeletal structures and dynamic movement simulator and other implementations such as AnyBody [2] are not designed with real-time execution as their focus.

Different attempts to bridge this gap and simulate validated movement description of musculoskeletal models in real-time exist, notably RTOSIM [3] and more currently OpenSimRT [4]. RTOSIM is a real-time implementation of OpenSim which solves inverse kinematics and dynamics at 2000fps using motion-capture data and measured ground reaction forces from force plates. OpenSimRT extends this work by including inverse kinematics from IMU input and model based Ground Reaction Force and Moments (GRFM) prediction for walking, as well as estimation of individual Muscle Activation and Joint Reaction loads.

Integration of multiple systems becomes a concern once a system starts to become more complex. ROS [5], widely used for real-time robotics research applications is a framework which tackles integration issues by introducing the concept of nodes, which execute a certain specific task and a communications framework based on pre-compiled messages published on topics and service calls which those nodes use to communicate with each other.

Pinocchio [6], a ROS integrated, real-time rigid multibody physics mechanics by the Gepetto Research Group offers the support for OpenSim models by using the ospi library. However it does so by parsing OpenSim models and simulating them with Pinocchio, instead of using OpenSim directly. An interface which uses OpenSim itself offers the advantage of using other modules from the feature-rich OpenSim platform.

OpenSimRT provides example tests which are almost equivalent to ROS nodes, however without a framework such as the ros_comm interface, integration and development of variation of nodes become cumbersome as the number of variations becomes large. Modifications of the underlying code become increasingly harder, specially when multiple different modes of operation (such as different sensor inputs or algorithms and changes in models) are considered.

A system with many components may become hard to reimplement on different hardware or operating systems, a term colloquially referred to as "dependency hell". To avoid such issues, we chose a pipeline using Docker [7], a solution often used by the AI community to mitigate replication issues[8]. Docker provides containerization, which allows for more control of installed libraries and communications, allowing containers to be shared online and a more quick deployment and testing. To allow for the system to continue to grow in complexity, we thus decided to use ROS, Docker and a customized version of OpenSimRT with ROS integration, which will present in this paper.

II. ARCHITECTURE DESCRIPTION

A. System overview

A Dockerized [3] version of OpenSimRT was developed using a ROS container image with OpenSim 4.1 and OpenSimRT based on the Continuous Integration [9] GitHub workflow from OpenSimRT. The OpenSimRT package was

*This work was supported by Digital Futures

[1]Frederico Belmonte Klein is with MoveAbility Lab, Dept. of Engineering Mechanics, Kungliga Tekniska Högskolan, KTH 100 44 Stockholm, Sweden frekle@kth.se

[2]Ruoli Wang is with MoveAbility Lab, Dept. of Engineering Mechanics, Kungliga Tekniska Högskolan, KTH, Stockholm Sweden ruoli@kth.se

[3]Docker image builder available in https://github.com/opensimrt-ros/docker-opensimrt.

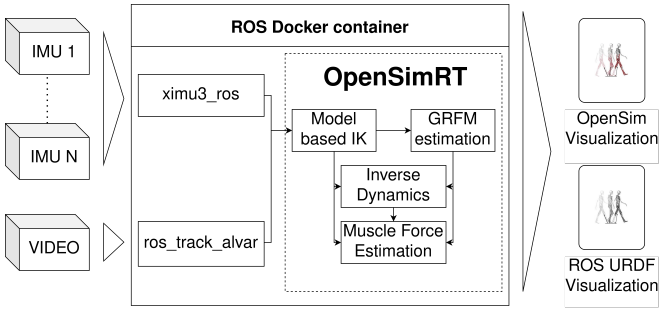


Fig. 1: Block diagram of the system showing OpenSimRT ROS package with its nodes. The ximu3_ros and ros_track_alvar provide rotation frame of reference transformations for XIMU3 and ALVAR markers respectively.

modified to be integrated with ROS[4] and suitable ROS messages were used. Nodes were altered to use ROS communications. An overview of the workflow can be seen in Fig. 1.

From the OpenSimRT project, we ported to ROS the following modules:

- Inverse Kinematics (IK) - reads orientation data from the subject and produces joint angles;
- GRFM estimation - provides ground reaction wrenches for normal walking by using a state-machine to detect current gait states using either virtual ground surfaces, or acceleration values of lower limb kinematics;
- Inverse Dynamics (ID) - solves ID by using a recursive Newton-Euler formulation with online filtering;
- Muscle Force Estimation - solves the static optimization muscle redundancy problem by minimizing muscle activations.

More details about the operation of the OpenSimRT nodes, can be found in Stanev et al [4].

B. Communication

Data transmission between nodes is done by ROS's middleware communications stack ros_comm, via network sockets, abstracted into topics, messages and services. The unit tests from OpenSimRT source-code provided a convenient interface as `TimeTableSeries` objects were adopted, transposed to a standardized `CommonTimed` ROS message[5] with varying length and column labels. To avoid transmitting redundant information, we separate metadata from contents and we used a service call to inform the labels. As every node requires this handshaking, it was implemented as a `CommonNode` from which other pipeline elements were derived. For topics which required more than one input, message_filters were used.

An additional type of message was also used to allow transferring not only positions, but also the velocities and accelerations. Derivatives are sensitive to noise, therefore joint position signal needs to be first filtered. To avoid having to filter joint positions multiple times in different nodes, this information is optionally computed only once and sent in a different type of message, the `PosVelAccTimed` message in addition to the normal `CommonTimed`.

C. Generation of the Unified Robot Description Format file

The lower body Unified Robot Description Format (URDF)[10] model generation was done based on the .osim model using Pinocchio and ospi2urdf library.[6]

The generated URDF model was visually adjusted on the origin (translation) of the knees joints and a custom joint publisher node was written.

D. Inertial Motion Unit interface

Inverse kinematics from IMU readings, originally in OpenSimRT as a custom driver, was replaced by a ROS TF interface[7]. This allows us to abstract IMU drivers and more easily replace orientation providers to use different IMU types or different sources of orientation input while the rest of the code remains unchanged. This change was also necessary as the IMUs used in the original OpenSimRT are no longer fabricated and communicated via oscpack.

As a demonstration, we implemented a ROS driver for the IMU system (XIMU3, x-io Technologies)[8]. Due to a different coordinate system and quaternion representation in OpenSim than ROS, a conversion is needed. The use of ROS TFs allows the use of augmented reality ALVAR markers[11] for a correct frame of reference (by attaching the marker to the IMU), which allowed us to separate the problem of getting the correct quaternion representation from the IMU and getting the correct transformation of quaternions (from the ALVAR input, only rotations are used) from ROS to OpenSim. ALVAR markers provide also a camera only solution, which may be sufficient in some cases.

III. RESULTS

Using one single orientation provider, such as either an AR Cube (see Fig.2) or a single IMU, it is possible to get the orientation of a single link. To use this information to test our implementation, rigid transformations were applied between all the links of the model, which allows us to test both the AR orientations and the IMU orientations of both the lower-limb model and the upper-limb model.

Using pre-recorded data, we were also able to test our URDF model against the internal visualization model from OpenSim (see Fig.3). In Fig.4, GRFM and muscle activation estimation were illustrated side by side with the URDF model. Note the GRFM model has a slight delay, because the URDF is still reading unfiltered joint angles.

[4]To reduce compilation times, OpenSimRT internals were separated from the package containing the ROS nodes.

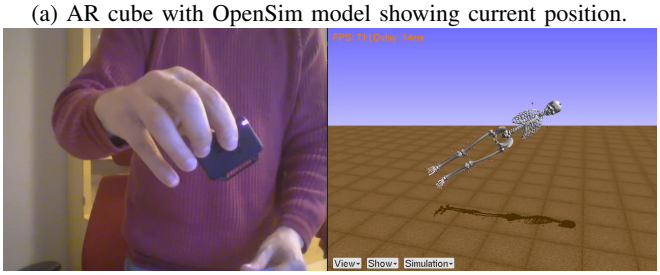
[5]An additional detail is the inclusion of an extra-field for timestamps in the message, necessary for slowed-down data playback from recorded data read from CSV or STO files, instead of using ROSBAGs.

[6]Included docker-opensimrt is the script `human_control.bash` which facilitates the installation of Pinocchio and ospi2urdf.

[7]ROS TF publishes rotation and translation transformations between frames of reference.

[8]Available at `https://github.com/opensimrt-ros/ximu3_ros`

(a) AR cube with OpenSim model showing current position.



(b) Using XIMU3 as TF source.

Fig. 2: Screencapture of real-time use of (a) AR cube and (b) XIMU3, showing both can provide orientation TFs.

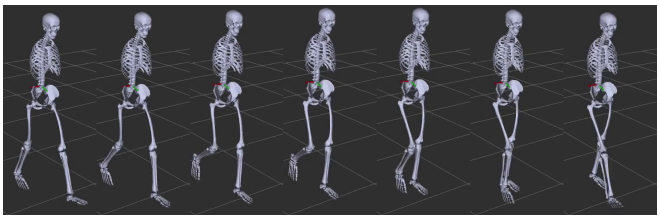
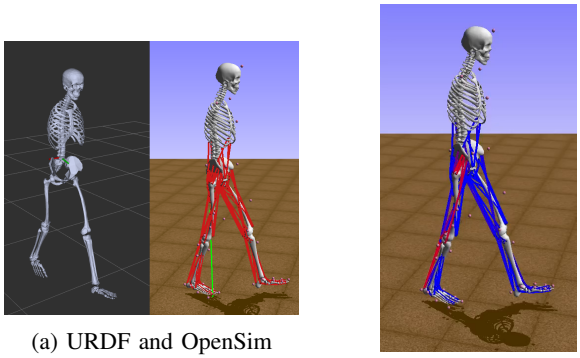


Fig. 3: Visualization of a gait trial by URDF model in rViz.

IV. DISCUSSION AND FUTURE WORK

This work aimed to allow easy replication, integration and extension of real-time human movement simulation for robotics and biofeedback applications. It provides fast model based IK via an agnostic interface allowing input from either AR markers or IMU orientation. Ground reaction wrenches can be estimated without additional sensor input for normal walking. Muscle activations can also be estimated, information which can be used for biofeedback or energetic cost estimation for exoskeleton control. Deployment with Docker is straightforward and if the focus of extension of this pipeline is on robotics alone, the whole implementation can be abstracted, limiting the required OpenSim knowledge. While the stack ROS-Docker-OpenSimRT presented offers all of these advantages, it is not without its own limitations.

A notable limitation is using a single IMU and pre-recorded data for testing the system. We reasonably assume that ROS TF static_transform_publishers emulate ximu3_ros TF publisher nodes, but multiple IMU sources were not tested. And while the models used were validated, this particular implementation was not, with further testing needed for the claim of validated human movement model to be made. Secondly, Docker makes ROS networking harder and integration with a robot (the main advantage of such a system), now



(a) URDF and OpenSim

(b) Muscle activation estimation.

Fig. 4: Side by side walk showing (a) OpenSim and URDF models. Green line is GRF prediction vector. On (b) muscle activation is shown, more intense red shading is more active.

requires more expertise. [9] Also, as it is, not all information is being published in appropriate ROS topics. The level of ROS integration, with a service handshake to initialize input labels and not the appropriate ROS messages[10] limits integration and visualization of topics in rViz. Finally, muscle activation estimation is slow, only reliable at <30fps[11], allowing only description of slow-walking.

Future work aims to tackle some of those limitations. Communication can be improved by using Nodelets (uses shared memory instead of sockets for communication) instead of normal nodes. Currently also Inverse Dynamics and Static Optimization are combined in a hybrid node to avoid multiple uses of message_filters, which is an additional source of lag. Nodelets can likely be used here with the same benefits and preserving a ROS design pattern. A planned extension is also to include ground reaction measurement with insole sensors to allow description of a wider range of movements. Further developments in this project aim to tackle these issues by increasing ROS integration, using Nodelets and parallelization.

V. CONCLUSION

This implementation of OpenSimRT in a ROS framework allows for wider integration efforts between biomechanics and robotics. It is extensible and allows validated description of human movement for real-time bio-mechanics applications, facilitating human robot interaction applications and enabling easier development of complex exoskeleton applications requiring ROS utilities.

ROS-OpenSim integration conveniently provides real-time human walking related parameters which should allow roboticists, clinicians, and researchers working on assistive robotics for human walking to measure those quantities for rehabilitation efforts, as well as more easily develop complex walking aid systems.

[9]The authors suggest real-time nodes be first written off-line reading `TimeTableSeries` entries from OpenSim 4.1 and then ported to ROS.

[10]Such as `geometry_msgs/WrenchStamped.msg` for GRFM.

[11]The machine used for testing is an Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz running Debian GNU/Linux 11 (bullseye).

REFERENCES

[1] S. L. Delp, F. C. Anderson, A. S. Arnold, P. Loan, A. Habib, C. T. John, E. Guendelman, and D. G. Thelen, "OpenSim: open-source software to create and analyze dynamic simulations of movement," *IEEE transactions on bio-medical engineering*, vol. 54, pp. 1940–1950, Nov. 2007.

[2] J. Rasmussen, M. Damsgaard, E. Surma, S. Tørholm, M. de Zee, and V. Vondrak, "AnyBody - a software system for ergonomic optimization," in *Fifth World Congress on Structural and Multidisciplinary Optimization*, Schönenfeld & Ziegler, Jan. 2003.

[3] C. Pizzolato, M. Reggiani, L. Modenese, and D. G. Lloyd, "Real-time inverse kinematics and inverse dynamics for lower limb applications using OpenSim," *Computer Methods in Biomechanics and Biomedical Engineering*, vol. 20, pp. 436–445, Mar. 2017. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/10255842.2016.1240789.

[4] D. Stanev, K. Filip, D. Bitzas, S. Zouras, G. Giarmatzis, D. Tsaopoulos, and K. Moustakas, "Real-Time Musculoskeletal Kinematics and Dynamics Analysis Using Marker- and IMU-Based Solutions in Rehabilitation," *Sensors*, vol. 21, p. 1804, Jan. 2021. Number: 5 Publisher: Multidisciplinary Digital Publishing Institute.

[5] 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," in *ICRA workshop on open source software*, vol. 3, p. 5, Kobe, Japan, 2009. Issue: 3.2.

[6] Gepetto Research Group - Movement of Anthropomorphic Systems, "Pinocchio." `https://stack-of-tasks.github.io/pinocchio/`, 2022. Accessed: 2022-09-16.

[7] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[8] M. Hutson, "Artificial intelligence faces reproducibility crisis," *Science*, vol. 359, pp. 725–726, Feb. 2018. Publisher: American Association for the Advancement of Science.

[9] P. Duvall, S. Matyas, and A. Glover, *Continuous integration: improving software quality and reducing risk*. Addison-Wesley Professional, first ed., 2007.

[10] Ioan Sucan and Jackie Kay, "urdf - ROS Wiki." `http://wiki.ros.org/urdf/`, 2009. Accessed: 2022-09-19.

[11] Scott Niekum, "ar_track_alvar - ROS Wiki." `https://wiki.ros.org/ar_track_alvar/`, 2012. Accessed: 2022-09-16.