2023 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
Tomar, Portugal – April 26-27, 2023

2023 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC) | 979-8-3503-0121-2/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICARSC58346.2023.10129643

Modeling and Realistic Simulation of a Dexterous Robotic Hand: SVH Hand use-case

Francisco M. Ribeiro
SYSTEC (DIGI2), ARISE & ECE Dept.
Fac. de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias,
4200-465 Porto, Portugal
fmribeiro@fe.up.pt

Tiago Correia
SYSTEC (DIGI2), ARISE & ECE Dept.
Fac. de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias,
4200-465 Porto, Portugal
tpcorreia@fe.up.pt

José Lima
CeDRI
Instituto Politécnico de Braganca
Campus de Sta Apolónia, Portugal
jllima@ipb.pt

Gil Gonçalves
SYSTEC (DIGI2), ARISE & ECE Dept.
Fac. de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias,
4200-465 Porto, Portugal
gil@fe.up.pt

Vítor H. Pinto
SYSTEC (DIGI2), ARISE & ECE Dept.
Fac. de Engenharia, Universidade do Porto
Rua Dr. Roberto Frias,
4200-465 Porto, Portugal
vitorpinto@fe.up.pt

***Abstract*—Recent developments in dexterous robotic manipulation technologies allowed for the design of very compact, yet capable, multi-fingered robotic hands. These can be designed to emulate the human touch and feel, reducing the aforementioned need for human expertise in highly detailed tasks. The presented work focused on the application of two simulation platforms - Gazebo and MuJoCo - to a use-case of a Schunk Five Finger Robotic Hand, coupled to the UR5 collaborative manipulator. This allowed to assess the relative appropriateness of each of these platforms.**

***Index Terms*—Robotic Hand; Realistic Simulation; Collaborative Robotics; Modeling; Control; ROS**

I. Introduction

The drive to increase efficiency in manufacturing industries is growing, as the market tends to be more and more competitive. Current technological progress has led to the expansion of automated tasks in industrial environments, however there are still manufacturing duties that need to be handled by skilled human operators, due to their high level of complexity. Automation of industrial manufacturing processes, through the use of robots designed to perform previously manual operations, is an area of development that has received high level of attention in the last decades.

Recent developments in dexterous robotic manipulation technologies allowed for the design of very compact, yet capable, multi-fingered robotic hands. These can be designed to emulate the human touch and feel, reducing the aforementioned need for human expertise in highly detailed tasks. Nevertheless, these are very expensive equipment that require special attention when designing and testing control algorithms, as this can cause damage and unnecessary wear to the components.

The present work focused on the application of a use-case - Schunk Five Finger Robotic Hand, coupled to the UR5 collaborative manipulator - to the Gazebo and MuJoCo simulation platforms. This allowed to assess the relative appropriateness of each of these platforms for the specific case considered in this work. In Section II, some details regarding motion planning and simulation tools in the literature are provided. Also, the working principles of the Schunk Five Finger Hand are detailed. The development work is shown in Section III explaining the tasks performed to achieve the desired functionality of the prototype. The results that were able to be extracted and their analysis are in Section IV. Finally, in Section V some final remarks and some future investigation directions are pointed out.

979-8-3503-0121-2/23/$31.00 ©2023 IEEE

II. Related Work

Since the safety of the equipment, as well as safety of nearby humans, is of high priority in the design of collaborative robotic systems, such as the one composed by a dexterous robotic hand attached to a manipulator, simulation tools can be used in order to test and validate control solutions. Therefore, these take a very important role in this area of robotics along with the tools that allow for the motion planning of a robotic system, both of which will be analysed in this section.

A. Motion Planning

To execute a desired movement with a robot such as the ones in which this work is focused upon, it is necessary to calculate its trajectory, i.e. to plan the movement.

The trajectory planning generates the input references for the low level controllers to follow and this will lead the manipulator to follow the planned movement. Note that a trajectory may define speed and acceleration besides the position reference.

Tools have been developed for the purpose of motion planning and execution and have reached a high level of maturity. MoveIt is a ROS library that is a state of the art tool for robot motion planning [1]. It is a very widely used software package

due to its open source license, and implements the latest technology for motion planning, manipulation, kinematics, control and navigation [2]. Another advantage of MoveIt is its integration with the rest of the ROS framework such as the Gazebo simulator and RViz visualization software.

ROS itself is an open source framework for robot programming. A collection of libraries and tools support the implementation of software for robot control making the development process easier and faster [3]. ROS has a built-in messaging system that works in a publish/subscribe pattern without the user needing to worry about the implementation of such communication middleware. This results in clear interfaces meaning that the same code can be used towards many different kinds of robots. Some other arguments to be made in favor of ROS are the excellent documentation, frequent maintenance of libraries and active community. However, the lack of backwards compatibility between ROS versions sometimes leads to difficulties to overcome during the development such as adjusting the Linux version used or even building some packages from source.

MoveIt can be configured to use a number of different inverse kinematic solvers. The pre-installed packages that perform this function are targeted at serial kinematic chains, so a different solution must be searched in order to solve motion tasks on more complex robots such as multi-fingered robotic hands. BioIK is an open-source software package fully integrated in ROS and MoveIt that is capable of solving inverse kinematics for robots with arbitrary kinematic trees [4]. This package allows for the description of motion goals using a combination of individual sub-goals, each with an associated weight that relates to its priority in the computation process. This empowers a user to specify, for example, the position and orientation of each one of the fingertips in a dexterous robotic hand in order to grasp an object, making it very interesting in the scope of this work.

The mentioned above RViz visualization software, as the designation suggests, is a 3D visualization tool with graphical interface part of the ROS ecosystem. It allows for the inspection of any robot model and its functionality can be expanded by means of plugins. As an example, with the MoveIt plugin commands can be sent directly from RViz to the MoveIt interface to request trajectory for a certain motion.

B. Robotic Simulation

Simulation is a very important tool in robotics, allowing tests to be executed without access to the physical robot. The usage of simulation before transitioning to real hardware carries benefits as there is no harm or wear to the hardware which can be very costly and also likely to happen in a phase where the user is quite unfamiliar with it or, in the case of RL training when the agent's behavior is hard to predict. One can also run a wide range of simulations in parallel which can be beneficial for learning based control approaches.

In [5] popular simulation environments are tested and compared. The authors recommend Gazebo when developing for simulation and real systems due to the homogeneous ROS connection that allows the control to be used in the virtual and real world with minimal changes. It also performs best in trajectory tracking tests similar to the use case presented in the present work. MuJoCo performs second best in this use case and is recommended for RL training due to its popularity in OpenAI Gym simulations [6]. As these two platforms show the most promising results they will be analysed further below.

1) Gazebo [7]: An open source status 3D simulator that is integrated in the ROS framework, which explains its popularity.

It is equiped with 4 different physics engines: Open Dynamics Engine (ODE) [8], Bullet [9], Dynamic Animation and Robotics Toolkit (DART) [10] and Simbody [11] allowing for an easy adaptation to the user's needs. Empowers roboticists to have as detailed of an environment as they desire as it can simulate every aspect of the real world from gravity to inertias and ever sensor noise. Similar to ROS, Gazebo also benefits from the attributes of open source products, having at its disposal both a dedicated community and effective help forums, as well as an active development.

Gazebo is reported to have numerical instability when simulating low inertia articulated bodies, as is the case of a robotic hand due to the small dimensions and weight of the finger links [12]. This may be a problem under the proposed application in this project, however that will be verified later.

2) MuJoCo: (Multi-Joint dynamics with Contact) [13] is a physics engine that makes it possible to develop, simulate, and visualise contact dynamics of multi-joint systems rapidly and accurately.

As the main focus of this simulator is performance, it is able to navigate very complicated operations from the computational standpoint particularly in highly complex systems with several contact points to be processed, as is the case of a multifingered robotic hand. Other arguments in favor of MuJoCo for use cases that are similar to the one presented here are it's capability to simulate soft, rope like objects or even complex particle systems [14], supports soft contacts natively and is built with multiple contacts in mind hence inverse dynamics are well defined even in this situation [13].

The major downside of MuJoCo for simulation of manipulation tasks is the lack of an inverse kinematics solver and path planning, unlike Gazebo which, with its integration with ROS, acquires these features [14].

MuJoCo models are written in the MJCF format which is an XML derivation, however Unified Robot Description Format (URDF) typically used in Gazebo and ROS can also be loaded natively or converted to MJCF if needed.

C. SCHUNK SVH 5-Finger Hand

The SCHUNK SVH 5-finger hand (S5FH) is an anthropomorphic robotic hand with a high degree of similarity to the human hand regarding its shape, size and general appearance. As a consequence of this similarity to human hand dimensions the number of motors within its structure had to be reduced. This is the solution to ensure the speed and strength

requirements while keeping the small size. Therefore the S5FH has 20 joints but only 9 DoF [15], [16].

1) Kinematic Model: Owing to the lower number of motors when compared to the joints, mechanical couplings are performed to obtain full control of the hand. This coupling is reached by means of mechanical transmissions between motors and joints that define mechanical synergies.

Let $\boldsymbol{m}$ be the vector of the 9 motor variables. Having $\boldsymbol{q}_0$ be the vector of offsets that represents joint angles corresponding to the central motor positions, the hand joints can be expressed using the following equation:

$$\boldsymbol{q} = \boldsymbol{S}_m \, \boldsymbol{m} + \boldsymbol{q}_0 \tag{1}$$

where $\boldsymbol{S}_m$ represents the matrix of the mechanical synergies.

Another approach to take into account these mechanical dependencies is to represent joint variables not in order to the motors but instead using as reference a main joint that connects to the same motor. This is done in [16] as it can be observed in the kinematic model representation in Figure 1. The multiplication factors used here can be extracted from the matrix $\boldsymbol{S}_m$ as the one presented in [17] by finding the relations between joints mechanically coupled to a common motor.

In spite of full detailed physical properties of the hand, such as size and mass of the individual components, not being provided by the manufacturer in the public documentation, it is known that the hand is in a 1:1 ratio to an average human hand and its weight is $1.3kg$ [17]. Some overall measurements are published in [17].

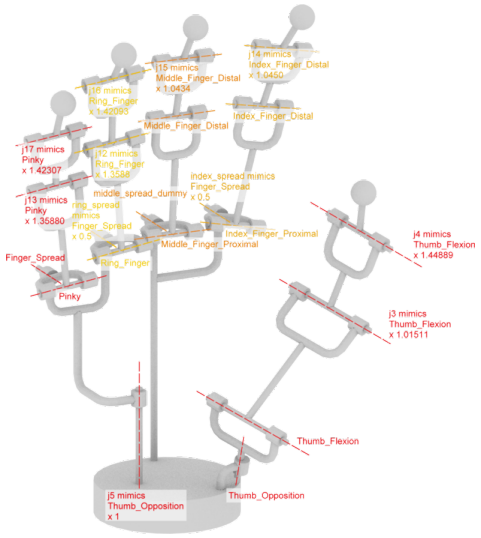


Fig. 1: Individual joint positions in the kinematic model [16]

2) Controller Description: The wrist integrates the electronic components used for the control of the hand. Embedded in it there is the Mecovis S5FH-Controller which has been designed to control the S5FH in particular. This controller contains motor drive circuits, sensor interfaces for position feedback, motion controller and a serial communication interface to a host [16]. A field-programmable gate array (FPGA) is used to implement the firmware's functionality wich provides high flexibility and performance.

Regarding the control itself, it is performed by cascading a position controller and a current controller. The position controller is implemented as a Proportional–Integral–Derivative (PID) controller with anti-windup. The reference signal for this controller to follow is set via the communication interface by writing to an internal register of the FPGA and its output is internally routed to the reference input of the current controller. The current controller is introduced in the form of a Proportional-Integral (PI) controller with anti-windup and its output is connected to a PWM generator that drives a motor.

The constants that define the behavior of the controllers described above are also written via the communication channel. The Robot Operating System (ROS) driver for the S5FH already contains the values and programmed protocols to send them as packets to the Mecovis. Hence, its usage when dealing with the real hardware streamlines the process of setting up.

III. Prototype

A. Gazebo Simulation

The important concepts regarding the Realistic Simulation on Gazebo will be presented in the current Section.

1) URDF Model: For kinematic computation and simulation purposes, a model that accurately represents the physical characteristics of the robot and its links and joints is of the upmost importance. Therefore, the first step of this work was the development of such model.

The ROS driver package for the Schunk multifingered hand [1] provides a bare-bone model using Unified Robot Description Format (URDF) and Xacro, which is a XML macro language. In the original state it is not very useful as there are a lot of components missing, for instance inertial properties and surface characteristics such as friction, being the latter essential when the objective is gripping an object. The finger links and most sections of the hand model can be well approximated by cylinders, hence a macro that could be used to introduce inertial parameters in the associated URDF file was developed. The models produced to execute the simulations as well as the code corresponding to the developed control pipeline can be found at https://github.com/franciscombr/SVH-simulation-suite.

Unfortunately the hand was not physically available for the measurement of the necessary parameters and the provided documentation is not detailed in this matter, not revealing the individual mass of each link neither their corresponding radius and length. It is predicted that in the future with access to the hardware these parameters can be extracted nevertheless, while that is not the case, some rough approximations have been used.

To make the model ready for Gazebo simulation some other information is required. Regarding contact properties, the static

[1]ROS Driver package for the SVH hand https://github.com/fzi-forschungszentrum-informatik/schunk_svh_driver, as of October 18, 2022

and kinematic friction coefficients are set to an ideal unitary value.

In order to use the ros_control package [18] it is also necessary to configure the transmissions which will effectively propagate position, velocity and effort variables between actuator and joint spaces. A simple transmission with a gearing ratio of 1:1 between actuator and joint is introduced for each one of the 9 DoF that the hand contains.

Gazebo itself does not provide support for such coupling of joints, yet this functionality can be added by means of a community-made Mimic Joint plugin [2]. This plugin couples a mimic joint to a parent joint and maintains a defined linear relationship between their angles set by a multiplier and an offset. It also has the capability of loading a PID controller in the mimic joint that would follow the angle of the parent joint as a reference, however this is not the case here as the joints are mechanically coupled in reality. A macro was created in order to simplify the usage of this plugin.

Although not being strictly necessary for the correct representation of system dynamics, a force feedback ground truth sensor was added to the tip of the index finger in order to extract data that is representative of the behavior of the body under contact. This sensor implementation is part of the Gazebo library of sensors in which it is referenced as F3D. A macro was defined to streamline the addition of the plugin to the URDF model.

2) MoveIt Configuration Package: Using a robot through MoveIt happens via a configuration package that needs to be created. For this, the framework includes a tool called *MoveIt Setup Assistant* that, through a comprehensive graphical user interface (GUI), allows for the creation of such configuration package.

The process starts by importing the URDF that includes the description of the robot. After this, planning groups, which are used to describe different parts of the robot, can be created. In this particular case, as we are using the BioIK kinematic solver that supports non-trivial kinematic chains, the entire robot including the manipulator and the dexterous hand can be added to the same group.

Fixed poses can be added to the configuration and two were created, namely ”home” and ”test”. The first one places the robot in a neutral position while the second is used to touch the indicator finger with a surface in order to observe contact performance of the simulator. Joint angles of both poses are detailed in Table I.

The next key step is to add passive joints. These joints are not directly actuated on the robot therefore the planner must be informed not to kinematically plan for these joints as they can not be controlled. All the passively actuated hand joints are specified in this step. Finally, controllers are added to the controllable joints of the robot in order to be able to interface with the interfaces defined in the URDF model, making them follow the position trajectory that is computed by the planner. The Setup Assistant is capable of doing this automatically using the ”Auto Add” function. The information in the configuration package is used to launch a move_group node that has the architecture presented in [1]. The move_group_interface is a C++ interface that provides easy access to the most important functionality of the node, such as planning trajectories and executing them, so it will be the main tool to interact with the robot from a high level.

[2]Gazebo plugins by the Robotics Group at the University Of Patras https://github.com/roboticsgroup/roboticsgroup_upatras_gazebo_plugins

TABLE I: Defined robot poses details

Joint	“home”	“test”
Finger_Spread	0	0
Index_Finger_Distal	0	0
Index_Finger_Proximal	0	0
Middle_Finger_Distal	0	0.6981
Middle_Finger_Proximal	0	0.4363
Pinky	0	0.4363
Ring_Finger	0	0.4363
Thumb_Flexion	0	0
Thumb_Opposition	0	0
shoulder_lift_joint	-1.5447	0.4014
shoulder_pan_joint	1.5707	1.5707
elbow_joint	1.5447	0.9250
wrist_1_joint	-1.5794	2.3387
wrist_2_joint	-1.5794	1.5794
wrist_3_joint	0	0

B. MuJoCo Simulation

The important concepts regarding the Realistic Simulation on MuJoCo will be presented in the current Section.

1) MJCF Model: The previously presented URDF model can be converted to the MJCF format. The first step is to convert all the mesh files that are provided in .dae to .stl as this is the only format supported by MuJoCo. Some compiler definitions can be added to the URDF. These will point to the directory that contains the meshes, indicate if we want the inertia matrices to be automatically balanced and if we want to discard the visual aspect of the robot and replace complex mesh files with simple convex hull geometry files, respectively. More compiler definitions can be found in the MuJoCo documentation.

Finally, the Xacro file with the macros can be converted to plain URDF and this resulting file can be compiled to MJCF by using the *compile.cc* tool that comes packaged with MuJoCo.

The pre-built *simulate.cc* tool packaged with MuJoCo can be used to test the generated model after compilation. Some things need to be added manually to the model to make it functional, namely the actuators and the equalities that represent the joint coupling. Actuators are mainly described by their type, name and the joint that they are coupled to. Some other details can be added to add higher degree of fidelity such as control range and gearing ratios.

In order to physically introduce the mechanical interdependence of the SVH hand joints, equality parameters are needed in the model. The equalities have a certain type, in this case they are joint equalities and need some configurations.

The ”polycoef” parameter is very important when describing equalities. This string sets the coefficients a_0 to a_4 for the quartic polynomial that describes the constraint. Therefore

the following equation applies between two joints associated together by an equality:

$$q_1 = a_0 + a_1\, q_2 + a_2\, q_2^2 + a_3\, q_2^3 + a_4\, q_2^4 \qquad (2)$$

2) MoveIt Configuration Package: Creating the MoveIt configuration package to interact with the simulator in MuJoco is very similar to the process followed previously for the Gazebo case. The only difference is that there is no need to setup controllers at this stage as they will be defined in the created virtual hardware interface described in III-B4.

3) Simulation Environment: Comparing to Gazebo, MuJoCo does not have native integration with the ROS framework. Therefore, in order to take advantage of the motion planning tools that ROS offers, a MuJoCo simulation environment with the capability of subscribing and publishing to ROS topics had to be developed.

The connector was developed in C++ and has the main objective of sending to MuJoCo joint reference positions associated with trajectories generated by MoveIt and ROS. MuJoCo on the other hand sends data to ROS containing current state of the robot, which is very important for motion planning, and also position data regarding any object in the scene that the user wants to keep track of.

The subscribing of joint reference positions runs asynchronously while the simulation is executed according to the flowchart presented in Figure 2.

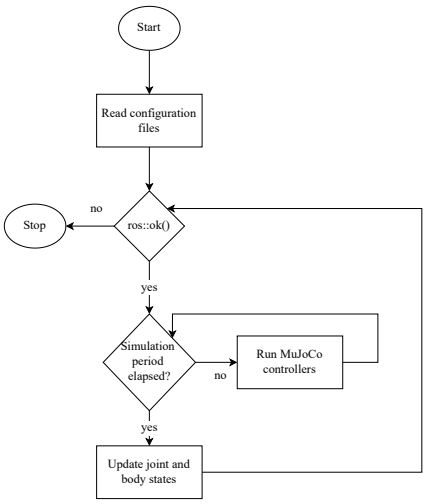


Fig. 2: Flowchart of the developed simulation environment

This environment can be easily configured to be used with another model as it is configured though a single *.yaml* file.

This provides the information to create a new object from the joint_ros_connector class including the name linked to the control topic, pose topic and initial pose topic as well as the proportional and differential gains for the MuJoCo controller responsible for the joint at hand. The controller function will be executed periodically at each simulation step adjusting the effort of each of the actuators in order to follow the joint position reference received. This function is implemented by a PD controller.

4) MuJoCo Virtual Hardware Interface: The simulation environment described above, although capable of communicating through ROS messages with other nodes, is limited to receiving joint reference position values for each of the registered articulations of the robot in the model. The issue is that the MoveIt motion planner produces full trajectories which are sequences of joint reference positions with a time instant associated to reaching them. The data frame in which this data in embedded and sent is quite complicated so a solution was found to avoid having to implement a parser to do this task and at the same time keeping the architecture as modular as possible, hence future proofing the system.

Taking the example of the inner working of ROS when communicating with a simulation in Gazebo, MoveIt sends the trajectory to the node running the ROS controllers. The controllers send position or effort goals, depending on their type, to a lower layer. This layer is composed by hardware interfaces that are compatible with the controllers and create a level of abstraction to the actuators associated to it. In a real implementation of a simple robotic system this hardware interface can receive for example an effort reference from the controller and send a motor current reference to the robot via a serial interface.

This architecture can be slightly modified and replicated in order to produce a Controller Interface that receives parsed joint position goals from the controllers and sends those same joint position goals to the MuJoCo simulation environment as it is represented in Figure 3.

The *ros_control* package includes many different kinds of controllers. In this case the needed function is only trajectory parsing and sending the position reference directly to the controller interface so Joint Trajectory Position Controllers are used. These are configured in a ”controllers.yaml” file in the Controller Interface package.

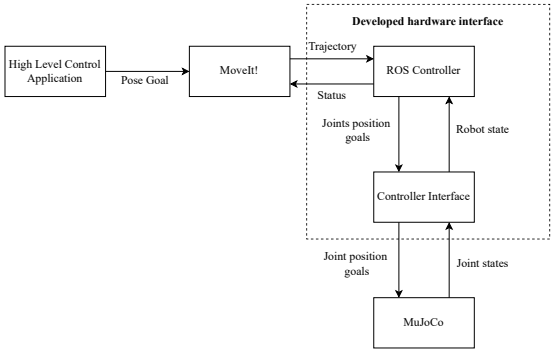


Fig. 3: Architecture of the system using the developed hardware interface

IV. Results

The described prototypes were used for the collection of data with the objective of comparing the performance of the studied simulator platforms. The observations are presented in this chapter and some inferences are drawn from them. Demos

of the robot in Gazebo and MuJoCo going from the "home" to "test" pose can be found in the following links: https://youtu.be/-_W3hOeciS0 and https://youtu.be/EMqRMzmRrfQ, respectively.

A. Gazebo Contact Behavior

Firstly, data regarding joint states of the robot, when the index finger touches the surface in Gazebo simulation was captured and some examples of joint positions are presented in Figures 5a and 6a. Also, using the force feedback sensor in the index finger, values of the vertical component of contact force are stored and displayed in Figure 4a. In both cases, data was recorded for three different settings of inertia of the finger links consisting of diagonal matrices of $0.001kg\,m^2$, $0.01kg\,m^2$ and $0.1kg\,m^2$.

The collected joint position data was analysed by computing the Mean Absolute Deviation (MAD) of the data with respect to the running average of itself, calculated with a window size of 10 samples. This statistical indicator, which can be calculated by the expression in 3, provides a sense of the degree of oscillation and instability of the reading.

$$MAD = \frac{\sum_i |x_i - \bar{x}_i|}{N} \tag{3}$$

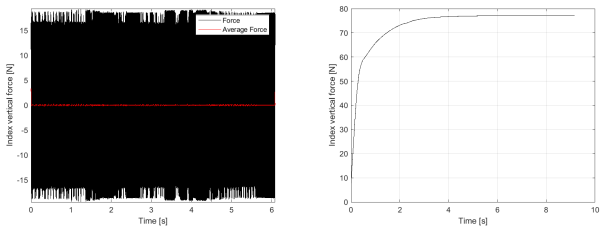
where N is the total number of samples, x_i is the i^{th} sample and $\bar{x}_i$ is the running average around the i^{th} sample. The value of this indicator for each joint is displayed in Table II.

TABLE II: Gazebo joint position dispersion assessment

Joint	MAD (inertia @ $0.001kg\,m^2$)	MAD (inertia @ $0.01kg\,m^2$)	MAD (inertia @ $0.1kg\,m^2$)
"Finger_Spread"	0.0240e-3	0.01541e-4	0.00111e-4
"Index_Finger_Distal"	0.0097e-3	0.00091e-4	0.00141e-4
"Index_Finger_Proximal"	0.0204e-3	0.00151e-4	0.00161e-4
"Middle_Finger_Distal"	0.0072e-3	0.00171e-4	0.01111e-4
"Middle_Finger_Proximal"	0.0196e-3	0.03041e-4	0.03261e-4
"Pinky"	0.0222e-3	0.02871e-4	0.03111e-4
"Ring_Finger"	0.0221e-3	0.03011e-4	0.03121e-4
"Thumb_Flexion"	0.0471e-3	0.01221e-4	0.00331e-4
"Thumb_Opposition"	0.4379e-3	0.05771e-4	0.00531e-4
"shoulder_lift_joint"	0.0151e-3	0.10011e-4	0.12251e-4
"shoulder_pan_joint"	0.0149e-3	0.02301e-4	0.01531e-4
"elbow_joint"	0.0173e-3	0.15331e-4	0.17551e-4
"wrist_1_joint"	0.0474e-3	0.34761e-4	0.38181e-4
"wrist_2_joint"	0.0603e-3	0.00921e-4	0.01131e-4
"wrist_3_joint"	0.0118e-3	0.06101e-4	0.04361e-4
Average	5.1799e-05	5.8178e-06	5.7914e-06

Interpretation of the results up to this point confirm that Gazebo has in fact numerical stability issues when simulating articulated bodies with small inertias under contact. Figures 5a and 6a show a very unstable behavior of the joint states. With the increase of inertia in the simulated articulated bodies the average MAD shows a reduction in its value which reflects the associated decrease in instability also visually evident in the graphical representations of the joint states.

The effect that the apparent vibration in the joints has over the force applied over an object, demonstrated in Figure 4a, is to be expected as the contact is not constant but intermittent.



(a) Gazebo. (b) MuJoCo.

Fig. 4: Graphical representation of index tip contact force for inertias of $0.1kg\,m^2$

High oscillation in the force is therefore observed leading to a moving mean of the contact force that is approximately zero. Application of normal forces to the surface of an object is absolutely essential for gripping and manipulation tasks as these create the friction required to move the object, hence Gazebo seems to not be a very adequate solution for the realistic simulation problem in the scope of this project.

B. MuJoCo Contact Behavior

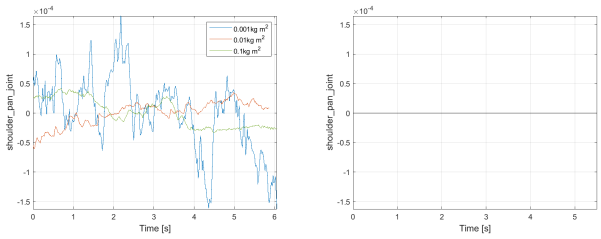
Similarly, for analysis of the MuJoCo realistic simulation environment, joint states of the robot when the index is in contact with a surface and the force that is exerted over said surface were captured. Time series of joint states are displayed in Figures 5b and 6b while force readings are in Figure 4b.

The same metodology used to process the states in the Gazebo case was also applied here, for which the results are presented in Table III.

TABLE III: MuJoCo joint position dispersion assessment

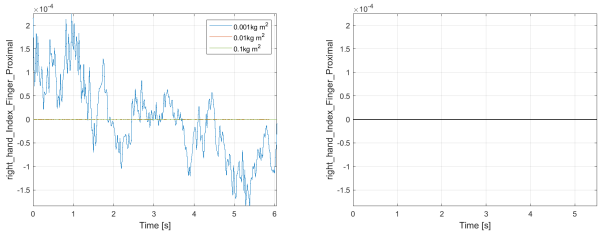
Joint	MAD (inertia @ $0.1kg\,m^2$)
"Finger_Spread"	0.0000
"Index_Finger_Distal"	0.0000
"Index_Finger_Proximal"	0.0000
"Middle_Finger_Distal"	0.0354e-15
"Middle_Finger_Proximal"	0.0555e-15
"Pinky"	0.0330e-15
"Ring_Finger"	0.0000
"Thumb_Flexion"	0.0000
"Thumb_Opposition"	0.0000
"shoulder_lift_joint"	0.0422e-15
"shoulder_pan_joint"	0.0845e-15
"elbow_joint"	0.1082e-15
"wrist_1_joint"	0.1400e-15
"wrist_2_joint"	0.1561e-15
"wrist_3_joint"	0.0001e-15
Average	4.3664e-17

MuJoCo presents a more stable operation that can be verified not only visually from Figures 5b and 4b, but also analytically from the results obtained. The joint states mostly stabilize in a final value when under contact which translates into an average MAD that is more that 10 orders of magnitude lower when compared to the equivalent test in Gazebo. The contact force exerted by the tip of the index finger shows a response much more similar to what was expected from a realistic simulation of the contact. Consequently, MuJoCo can



(a) Gazebo. (b) MuJoCo.

Fig. 5: Graphical representation of shoulder_pan_joint position



(a) Gazebo. (b) MuJoCo.

Fig. 6: right_hand_Index_Finger_Proximal position in graphical representation

be considered a valid candidate to perform simulation of the robotic system that is the protagonist to this project, when applied to grasping and manipulation tasks.

V. Conclusion and Future Work

In this work, two robotic simulation tools were explored regarding their adequacy for realistic simulation of a dexterous robotic hand and manipulator system for highly detailed tasks. The aforementioned exploration involved the development of description models compatible with Gazebo and MuJoCo, as well as support applications for the latter. The results allow to conclude that Gazebo, in spite of its higher user-friendliness and integration with the ROS framework, is not well adapted to use cases such as the one presented in this work. MuJoCo shows promising results in the tested setting and, through the developed virtual hardware interface, the same benefits of ROS integration can now also be taken advantage of when using this alternative simulation platform. The produced models, as well as the code corresponding to the developed control pipeline can be found at https://github.com/franciscombr/SVH-simulation-suite.

Future work on this subject can include the realistic physical parameters' modelling of the SVH hand, once the hardware becomes available. This will allow to reach simulation with a higher degree of fidelity to reality. Another step can be exploring the capabilities of the MoveIt motion planner as well as control algorithms in the literature. The transition to this process will now be faster due to the developed simulation pipeline in the course of this work.

Acknowledgment

The authors acknowledge the support of R&D Unit SYSTEC Base (UIDB/00147/2020) and Programmatic (UIDP/00147/2020) and the ARISE Associated Laboratory (LA/P/0112/2020), as well as the support of projects: *Digitalização da Arte Humana (Cibertoque)*, with reference POCI-01-0247-FEDER-072627, co-funded by FEDER, through COMPETE 2020 and Next-Gen Quality Control IoRT System with reference POCI-01-0247-FEDER-072616, co-funded by FEDER, through COMPETE 2020.

References

[1] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," 04 2014.

[2] "Moveit motion planning framework." [Online]. Available: https://moveit.ros.org/

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "Ros: an open-source robot operating system," vol. 3, 01 2009.

[4] P. Ruppel, N. Hendrich, S. Starke, and J. Zhang, "Cost functions to specify full-body motion and multi-goal manipulation tasks," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3152–3159.

[5] M. Körber, J. Lange, S. Rediske, S. Steinmann, and R. Glück, "Comparing popular simulation environments in the scope of robotics and reinforcement learning," 2021. [Online]. Available: https://arxiv.org/abs/2103.04616

[6] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Z. Openai, "Openai gym," 6 2016. [Online]. Available: https://arxiv.org/abs/1606.01540v1

[7] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154 vol.3.

[8] E. Drumwright, J. Hsu, N. Koenig, and D. Shell, "Extending open dynamics engine for robotics simulation," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6472 LNAI, pp. 38–50, 2010.

[9] E. Coumans, "Bullet physics simulation," in *ACM SIGGRAPH 2015 Courses*, ser. SIGGRAPH '15. New York, NY, USA: Association for Computing Machinery, 2015. [Online]. Available: https://doi.org/10.1145/2776880.2792704

[10] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, p. 500, 2018.

[11] M. Sherman, A. Seth, and S. Delp, "Simbody: Multibody dynamics for biomedical research," *Procedia IUTAM*, vol. 2, pp. 241–261, 12 2011.

[12] J. M. Hsu and S. C. Peters, "Extending open dynamics engine for the darpa virtual robotics challenge." [Online]. Available: http://osrfoundation.org

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

[14] J. Collins, S. Chand, A. Vanderkop, and D. Howard, "A review of physics simulators for robotic applications," *IEEE Access*, vol. 9, pp. 51 416–51 431, 2021.

[15] S. W. Ruehl, C. Parlitz, G. Heppner, A. Hermann, A. Roennau, and R. Dillmann, "Experimental evaluation of the schunk 5-finger gripping hand for grasping tasks," in *2014 IEEE International Conference on Robotics and Biomimetics (ROBIO 2014)*, 2014, pp. 2465–2470.

[16] "schunk_svh_driver - ros wiki," accessed: 2022-11-02. [Online]. Available: http://wiki.ros.org/schunk_svh_driver

[17] F. Ficuciello, A. Federico, V. Lippiello, and B. Siciliano, "Synergies evaluation of the schunk s5fh for grasping control," *Springer Proceedings in Advanced Robotics*, vol. 4, pp. 225–233, 2018.

[18] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017.