University of Duisburg-Essen
Faculty of Engineering
Chair of Mechatronics

Highly-Dynamic Movements of a Humanoid Robot Using Whole-Body Trajectory Optimization

Master Thesis

Maschinenbau (M.Sc.)

Julian Eßer
Student ID: 3015459

First examiner	Prof. Dr. Dr. h.c. Frank Kirchner (DFKI)
Second examiner	Dr.-Ing. Tobias Bruckmann (UDE)
Supervisor	Dr. rer. nat. Shivesh Kumar (DFKI)
Supervisor	Dr. Carlos Mastalli (University of Edinburgh)
Supervisor	Dr. Olivier Stasse (LAAS-CNRS)

September 18, 2020

UNIVERSITÄT DUISBURG ESSEN
Offen im Denken

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH
Robotics Innovation Center

Declaration

This study was carried out at the Robotics Innovation Center of the German Research Center for Artificial Intelligence in the Advanced AI Team on Mechanics & Control.

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

Bremen, September 18, 2020

Julian Eßer

Abstract

In order to further close the gap between robots and their natural counterparts, current research is driving towards exploiting the natural dynamics of robots. This requires a rethinking about the way we control robots for moving in a more dynamic, efficient and natural way.

Dynamic bipedal locomotion is a challenging problem and remains an open area of research. A key characteristic is the decoupling between the center of mass and the multi-body dynamics. Particular difficulties arise from effective underactuation, the mechanism complexity, as well as nonlinear and hybrid dynamics.

A common approach is to decompose this problem into smaller sub-problems that are solved sequentially. Many state-of-the-art frameworks rely on trajectory optimization based on reduced centroidal dynamics, which is transferred to a whole-body trajectory via feedback linearization using Inverse Kinematics (IK) or Inverse Dynamics (ID). Recent research indicates that solving this mapping with a local optimal control solver, namely Differential Dynamic Programming (DDP), instead of IK/ID, produces more efficient motions, with lower forces and impacts.

This master thesis contributes to the research field of dynamic bipedal locomotion by applying, evaluating and extending DDP-based whole-body trajectory optimization, pursuing three objectives: First, we develop an approach to constrain the optimal control problem allowing DDP to produce inherently balanced motions. Second, we evaluate our approach for quasi-static and dynamic motions in a real-time physics simulation and in real-world experiments on the lightweight and biologically inspired RH5 humanoid robot. Third, we examine the limits of the derived whole-body planning approach and the system design via solving highly-dynamic movements.

Keywords: Humanoid Robots, Dynamic Bipedal Walking, Motion Planning, Multi-Contact Optimal Control, Differential Dynamic Programming, Whole-Body Trajectory Optimization

Contents

Acronyms

OC Optimal Control

TO Trajectory Optimization

CHAPTER 1

Online Stabilization of the Planned Motions

This chapter presents a two-step validation of the physical compliance of the planned motions where an online stabilization is applied to track the optimal trajectories. At first, we proof the stability of the motions in a real-time physics simulation. Following this, we explore the feasibility of the motions in real-world experiments on a full-size humanoid robot.

1.1 Validation in Real-Time Physics Simulation

This section investigates the stability of the planned motions by applying an online stabilization based on joint space position control to track the trajectories obtained from Optimal Control (OC). To begin with, an overview of the simulation setup is given, then the tracking results of the planned motions are discussed.

1.1.1 Simulation Setup

The optimal motions are tested in the dynamical simulation environment PyBullet [1]. PyBullet is an open-source framework for robotics simulation that allows fast computation of rigid-body dynamics along with collision detection. The focus of PyBullet is to minimize the simulation-to-reality ('sim-to-real') gap, which is, despite constantly improving robot models, still a major problem in real-world experiments. To this end, the simulation in the simulator is setup in a similar way the motions would be tested o a real robot, namely interpolating the trajectories and closing the loop on joint position level.

The control loop on the real system (see Section 1.2) is running at a frequency of 1000Hz. The generated bipedal walking variants and highly-dynamic movements presented in ??–?? are generated with with an discrete OC formulation at 30Hz and 10Hz, respectively. To this end, the optimal trajectories are interpolated with a cubic spline in order to realize an up-scaling of the reference data to 1000Hz.

The control architecture consists of a simple PD-controller on joint space level. The controller is supposed to track both position and velocity reference trajectories with the standard PyBullet parameters in a real-time loop running at 1000Hz. In this real-time simulation, the same URDF robot model is used as in the Crocoddyl framework, including similar maximal motor torques. Furthermore, the parameters of the rigid contact models have been aligned between both frameworks to ensure comparability.

1.1.2 Motion Tracking With Joint Space Control

Following up, we investigate the capabilities of the presented control architecture for tracking the planned motions. To this end, we study the control tracking performance for the dynamic walking gait (??) and a highly-dynamic forward jump (??).

Dynamic Walking

To begin with, we analyze the motion tracking for dynamic walking gait. Fig. 1.1 shows the tracking performance of the joint level control architecture. The reference trajectories from OC are visualized as solid lines, while the resulting trajectories of the real-time physics simulation are shown as dotted lines. It can be seen that the controller follows the optimal trajectories fairly good. Small deviations can be seen around two seconds, which accounts for the lift off phase for the second robot step. This effect can be explained by the apparent abrupt change in the joint space, but is found to be marginal for the overall tracking performance.

As introduced previously, the control architecture is solely based on joint space position and velocity level. Although the tracking performance is good, this does not proof for the stability of the motions. Similar to the definition of robot tasks, also the evaluation of the motions should be pursued in task space. To this end, Fig. 1.2 monitors the according motion of the floating base. As can be seen, the floating base deviates about $\pm$ 10mm in x and y-direction , as well as + 5mm in z-direction.

In this context, it is important to notice that no controller tracks these task space quantities. Instead, they are merely the result of the joint space control performance. The largest deviations in task space occur during the first step and the according impact in the first two seconds of the motions. It becomes evident that these task space errors do not correlate to the peaks discussed on joint space level. Furthermore, one can observe oscillations in the stabilization phase at the end of the motion. This effect can be attributed to the fact that the systems slightly starts to swing after the

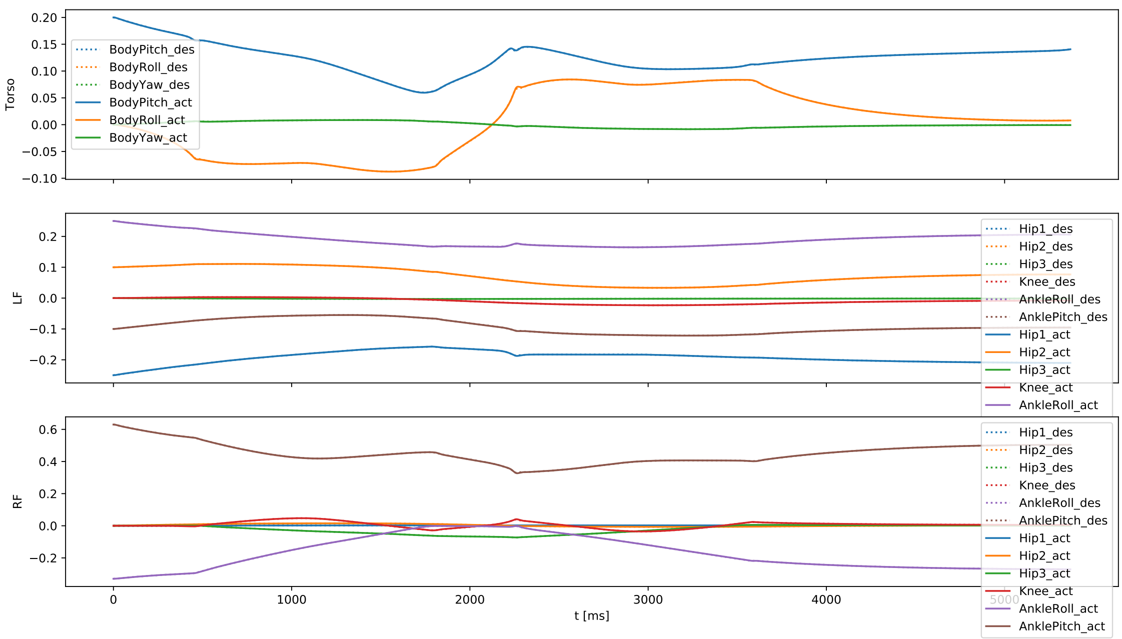


Figure 1.1: Control tracking performance on joint level for the dynamic walking gait.

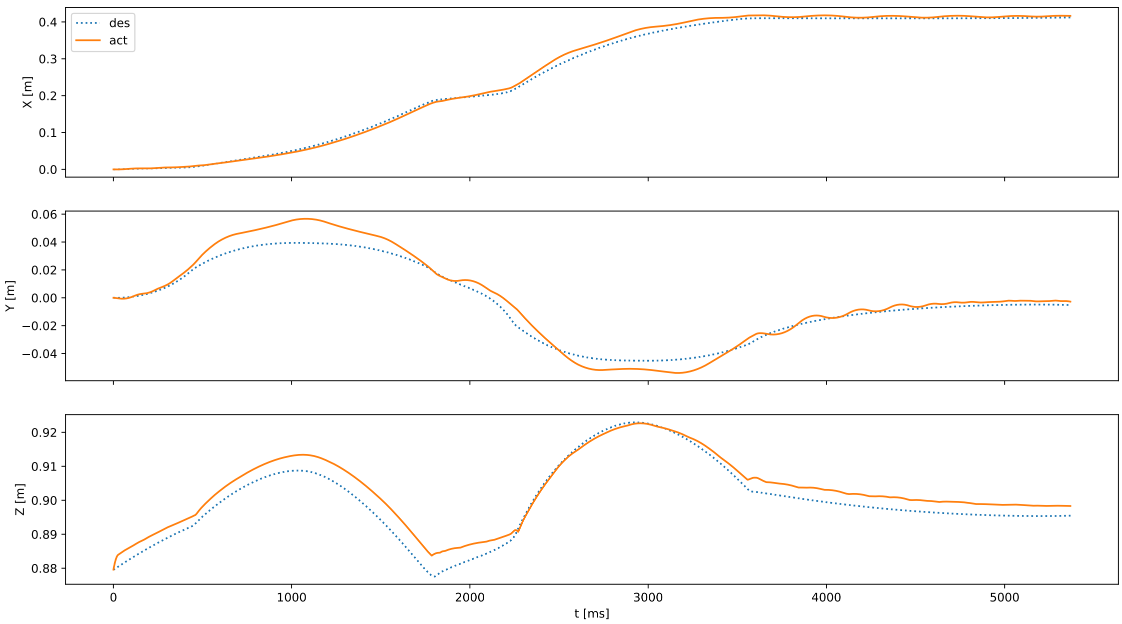


Figure 1.2: Motion of the floating base resulting from joint space control for the dynamic walking gait.

second impact. This behavior could be compensated e.g. with a dedicated control in task space instead of joint space.

Overall it can be stated that the dynamic walking motion can be successfully stabilized by the proposed control approach.

Forward Jumping

To identify the limits of the control approach, we now study the online stabilization for a highly-dynamic forward jump (??). In addition to the previously discussed dynamic walking, the jumping task introduces new challenges for the controller in terms of speed and robustness that will be investigated in the following.

1.3 shows the tracking performance of the joint level control architecture for the forward jump. In contrast to the case of dynamic walking, the joint space controller reveals larger tracking deviations. This is especially true for the most dynamic part of the motion, namely the acceleration of the base and finally the takeoff around 300-400ms. Large deviations can be especially seen for body pitch, and the knee joints, which turned out to be crucial for highly-dynamic movements.

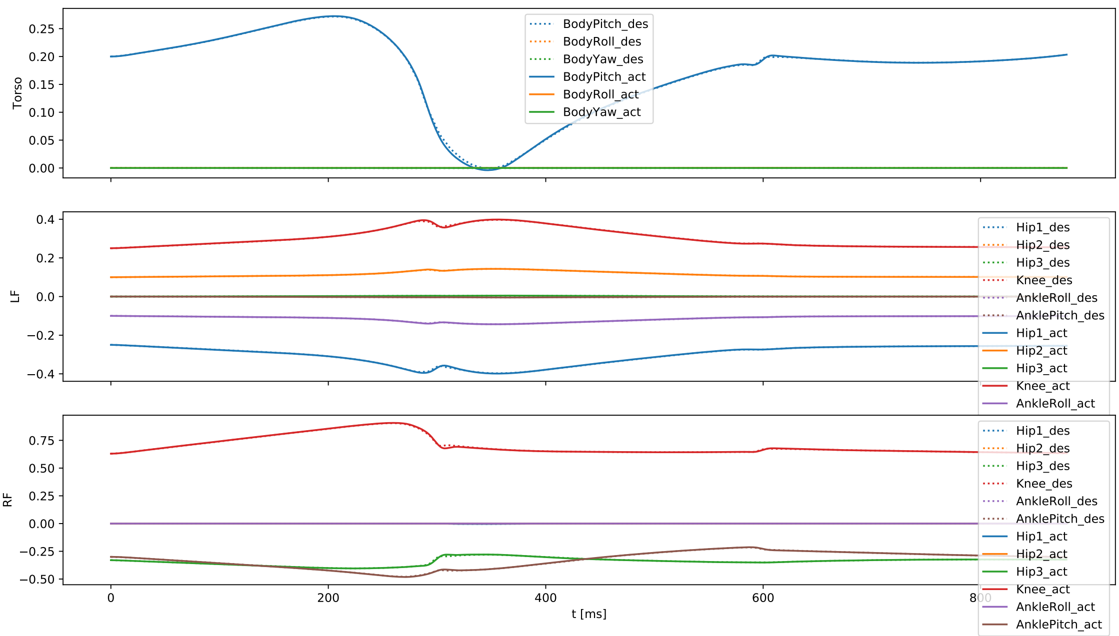


Figure 1.3: Control tracking performance on joint level for a forward jump.

In company with these findings, Fig. 1.4 monitors the motion of the (uncontrolled) floating base resulting from the joint space control performance. As becomes evident, the deviations in tasks space are also much higher compared to the dynamics walking case. While the height of the floating base is reasonable, the x-position shows tracking errors of about $\pm$ 5cm. Errors of this magnitude inevitably lead to instability of the movement to be performed. In this case, the strong deviation in task space causes the robot to tilt around the rear edge of the foot after the touchdown.

Consequently, it turned out that a mere control on a joint space basis is not sufficient to track highly dynamic movements, but is indeed appropriate to stabilize a dynamic walking gait in real-time.

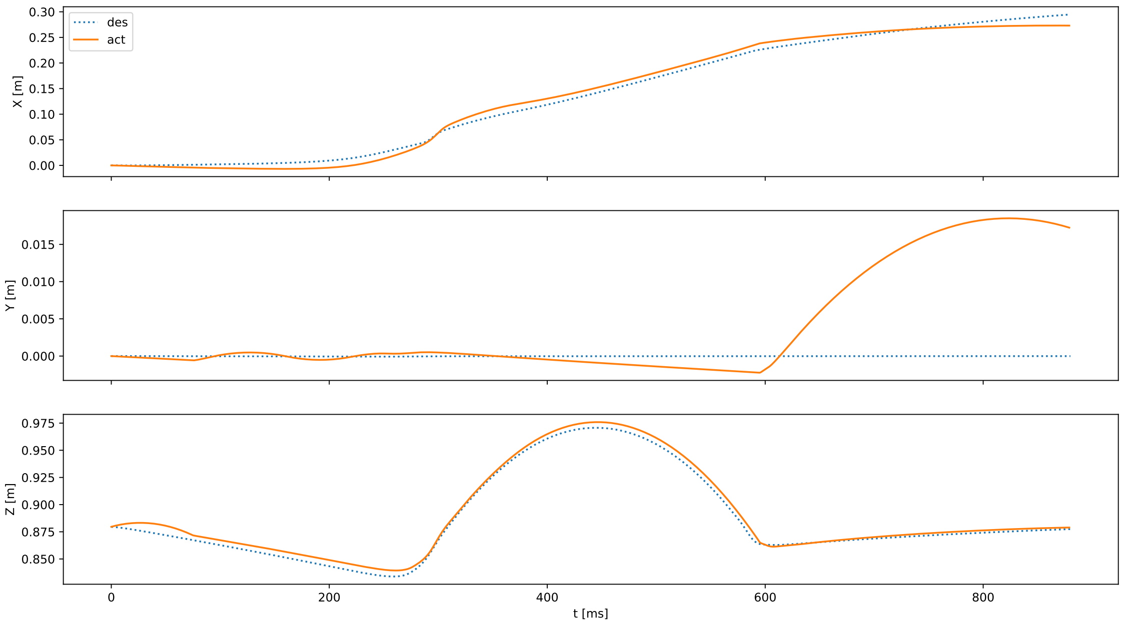


Figure 1.4: Motion of the floating base resulting from joint space control for a forward jump.

1.2 Validation in Real-World Experiments

This section presents experimental results of the planned motions on the full-size humanoid robot RH5 **??**. Analogously to the validation in PyBullet, the goal is to track the OC trajectories with an joint space online stabilization on the real system.

Based on the simulation results from the last section, highly dynamic movements are not evaluated on the real system in the context of this thesis. As discussed in the last section, these motions require more advanced control algorithms in task space due to the dynamic nature of the movement.

The rest of this section is structured as follows. First, an overview of the experimental setup with the involved components is presented. Following up, we verify the consistency between the used frameworks. Finally, four experiments are presented, gradually incorporating a new level of difficulty to evaluate the tracking performance of the online stabilizer (see Table 1.1).

1.2.1 Experimental Setup

An overview of the experimental pipeline is given in Fig. 1.5. The foundation for the experiments are the motion data generated offline with the proposed whole-body Trajectory Optimization (TO). The planned motions are then tracked in real-time with a joint space online stabilization on the real system. In the following, details on the involved components are provided.

The presented motion planning approach computes inherently balanced motions that are concisely captured in an appropriate file. This trajectory file contains the

Table 1.1: Overview about the difficulty of the conducted experiments.

	Balancing	Static Walk	Fast Squats	Dynamic Walk
Surface Contacts	✓	✓	✓	✓
Base motion	✓	✓	✓	✓
Swingleg motion	✓	✓	✗	✓
Step sequence	✗	✓	✗	✓
Impacts	✗	✓	✗	✓
Dynamic forces	✗	✗	✓	✓
Flight-phases	✗	✗	✗	✗
Success	✓	(✓)	✓	✗

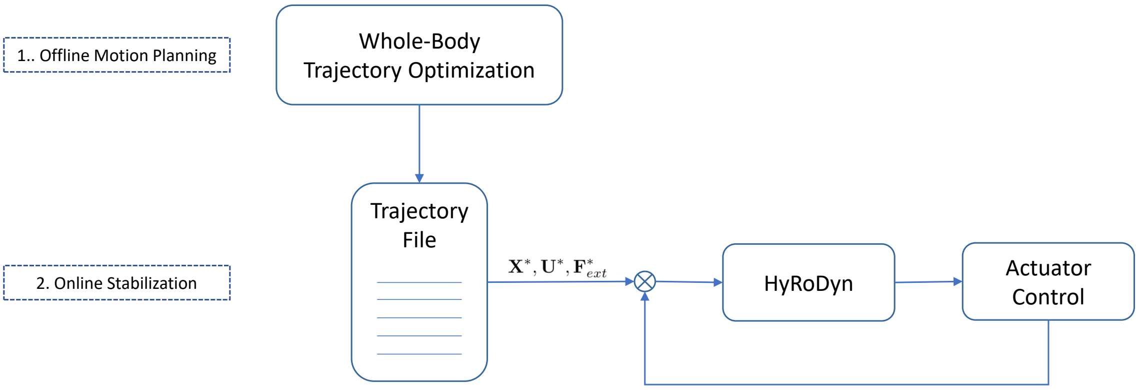


Figure 1.5: Overview about the experimental pipeline.

optimal state trajectories $\boldsymbol{X}^* = [\boldsymbol{q}^*, \boldsymbol{v}^*]$, optimal control inputs $\boldsymbol{U}^*$ and the resulting contact wrenches $\boldsymbol{F}^*_{ext}$ acting on the feet. In order to minimize the computational effort in the real-time loop, the file already encompasses data discretized to the desired frequency of 1000Hz. As in the PyBullet validation, the trajectories are interpolated using cubic splines in order to ensure smoothness an derivability.

As introduced in **??**, the novel RH5 humanoid robot contains multiple parallel mechanisms in order to achieve a high dynamic performance, superior stiffness and payload-to-weight ratio. This leads to the presence of various closed loops in series-parallel hybrid robotic systems, which are difficult to model and control. Most multi-body dynamics libraries, e.g. RBDL [2] and OpenSim [3], these loop closure constraints are solved numerically. HyRoDyn (Hybrid Robot Dynamics), is a recently presented modular software framework for solving the kinematics and dynamics of these type of series-parallel hybrid robots analytically, leading to improved accuracy and computational performance [4].

The planned motions are computed based on a serialized robot model. For dynamic real-time control, using this simplified model neglecting the closed loops turns out to be sufficient, although the accuracy is reduced [5]. However, the problem remains on transforming the results from the independent joint space, to the actuation space. In the context of this thesis, HyRoDyn is used to map the trajectories generated for the serialized robot model to compute the forces of the linear actuators.

HyRoDyn, as well as the low-level control of the RH5 robot is implemented using the Robot Construction Kit (Rock) middleware [6], which is based on the Orocod Real Time Toolkit. Low-level actuator controllers are utilized to compensate deviations from the reference trajectories. Analogously to the PyBullet validation, this control approach uses a cascaded position, velocity with an additional current control loop.

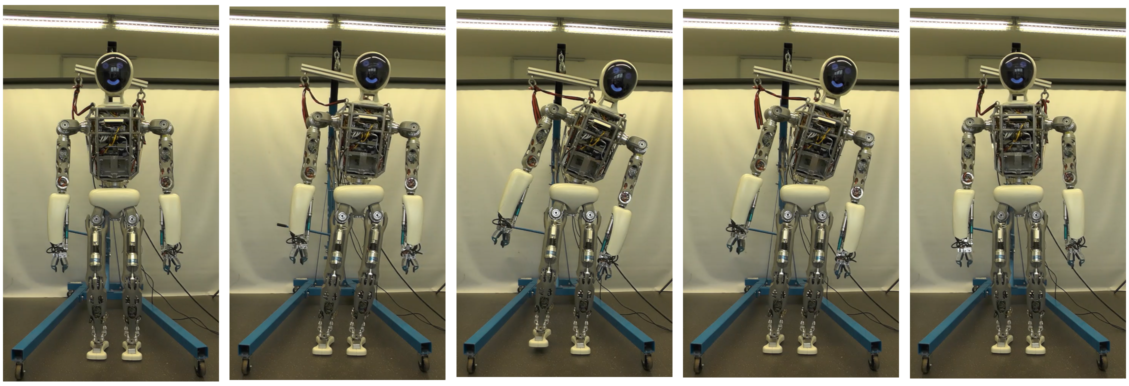
1.2.2 Consistency Between the Frameworks

Consistency of the frameworks involved in the motion planning and control pipeline is an indispensable prerequisite for the following up experiments. Hence, we will provide a brief proof of concept by checking the notability compliance of HyRoDyn and Pinocchio [7], which is used inside Crocoddyl for computation of robots dynamics and their analytical derivatives.

Recomputing the Contact Forces

CoM and ZMP Trajectories

1.2.3 Experiment I: One-Leg Balancing



a b c d e

Figure 1.6: Experiment No.1: One-Leg Balancing.

1.2.4 Experiment II: Static Walking

1.2.5 Experiment III: Fast Squats

1.2.6 Experiment IV: Dynamic Walking

CHAPTER 2

Conclusion and Outlook

2.1 Thesis Summary

2.2 Future Directions

Bibliography

[1] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[2] Martin L Felis. Rbdl: an efficient rigid-body dynamics library using recursive algorithms. *Autonomous Robots*, 41(2):495–511, 2017.

[3] Scott L Delp, Frank C Anderson, Allison S Arnold, Peter Loan, Ayman Habib, Chand T John, Eran Guendelman, and Darryl G Thelen. Opensim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, 54(11):1940–1950, 2007.

[4] Shivesh Kumar, Kai Alexander von Szadkowski, Andreas Müller, and Frank Kirchner. Hyrodyn: A modular software framework for solving analytical kinematics and dynamics of series-parallel hybrid robots. In *Proceedings of the International Conference on Intelligent Robots and Systems (Late Breaking Poster), Madrid, Spain*, pages 1–5, 2018.

[5] Shivesh Kumar, Julius Martensen, Andreas Mueller, and Frank Kirchner. Model simplification for dynamic control of series-parallel hybrid robots-a representative study on the effects of neglected dynamics shivesh. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5701–5708. IEEE, 2019.

[6] Sylvain Joyeux. Rock: the robot construction kit, 2013.

[7] Justin Carpentier, Guilhem Saurel, Gabriele Buondonno, Joseph Mirabel, Florent Lamiraux, Olivier Stasse, and Nicolas Mansard. The pinocchio c++ library: A fast and flexible implementation of rigid body dynamics algorithms and their analytical derivatives. In *2019 IEEE/SICE International Symposium on System Integration (SII)*, pages 614–619. IEEE, 2019.