2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)
Beijing, China, November 6-9, 2018

CARDSFlow: An End-to-End Open-Source Physics Environment for the Design, Simulation and Control of Musculoskeletal Robots

Simon Trendel[1], Yin Pok Chan[2], Alona Kharchenko[1,3], Rafael Hostettler[1,3], Alois Knoll[3], Darwin Lau[2]

***Abstract*— Motivated by the similar structure and actuation mechanism to humans and animals, the study of musculoskeletal robots has gained attention in recent years. However, the unilateral actuation property of muscles and high complexity of the mechanical system imposes great challenges on the design and control of such robots. An open-source realistic simulation platform for theoretical testing would therefore be advantageous for the research community of musculoskeletal robots. In this paper, an end-to-end open-source framework for the design, simulation and control of the general class of musculoskeletal robots (*CARDSFlow*) is presented. The framework consists of three advantageous features: 1) 3D computer-aided designs of musculoskeletal robots can be automatically converted for use with Gazebo and CASPR; 2) realistic physics simulation of musculoskeletal robots within Gazebo; and 3) integration of CASPR cable-driven robot controllers with CARDSFlow through the ROS platform. Simulation results on a two-link planar robot and the Roboy robot arm are presented to demonstrate the convenience to design and simulate musculoskeletal robots using CARDSFlow.**

I. Introduction

Bio-inspired robots, such as those inspired by humans and animals, have been studied since the beginning of the development of robots [1]. This ranges from those with bio-inspired functionality, such as flying [2], swimming [3] and walking [4], to humanoids with an anthropomorphic joint and rigid body structure [5]. More recently, the development of robots actuated by muscles (*musculoskeletal robots*) have been motivated by its higher end-effector to weight ratio [6], actuation similarities with musculoskeletal systems [7], expected better ability to produce human-like motion and a platform for the study of neuromuscular control.

Typically, musculoskeletal robots are actuated by: 1) *pneumatic artificial muscles* [8], for example, the Lucy robot [9]; and 2) *cables*, for example, the Roboy robot [10] (Figure 1), BM-Arm [11], and Kengoro robot family [12]. The key characteristic of such types of actuators is that they can only provide actuation in tension but not compression, resulting in the need of agonist and antagonist muscles (*actuation redundancy*). Furthermore, to maintain the compact structure of musculoskeletal systems, the actuators may wrap around the bone structure. Consequently, this creates a range of difficulties and challenges in the modelling [13], analysis [14]–[16] and control [17] of such systems.

[1]S. Trendel, A. Kharchenko and R. Hostettler are with the Roboy Project, Devanthro UG, Munich, Germany st@roboy.org, ak@roboy.org, rh@roboy.org
[2]Y. P. Chan and and D. Lau are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong ypchan@mae.cuhk.edu.hk, darwinlau@cuhk.edu.hk
[3]A. Kharchenko, R. Hostettler and A. Knoll are with the Department of Robotics and Embedded Systems, Technical University of Munich, Germany Alona.kharchenko@tum.de, {hostettl,knoll}@in.tum.de

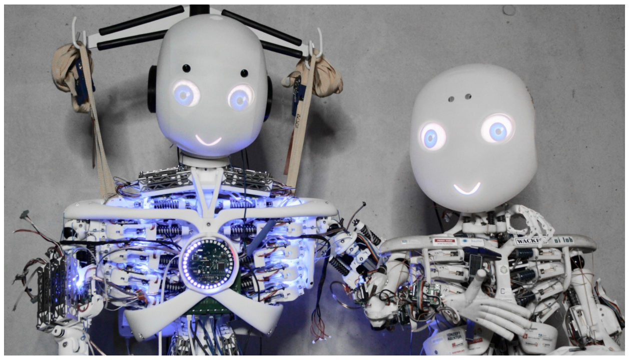


Fig. 1: Roboy cable-driven musculoskeletal robots

In addition to the challenges caused by the actuation properties, the increased complexity of the mechanical system makes it difficult to realize musculoskeletal robots in hardware. As such, the modelling and testing of algorithms in simulation provides enormous benefits in the advances of musculoskeletal robots. For example, the simulation of the inverse dynamics [13] allows the required muscle forces to produce particular motions to be determined such that the actuators and muscle locations can be designed. Furthermore, the control of the operational space of musculoskeletal robots [18] is a difficult theoretical problem and the testing in a realistic simulation environment should be performed before the deployment on the hardware to ensure safe operations.

Existing musculoskeletal robot projects, such as the Roboy or Kengoro, typically develop their own simulation and visualization software. Since the simulators are specific only to their developed robots, it cannot be easily used to study other musculoskeletal robots to design different systems. Moreover, these software are not easily accessible and inconvenient to use to test different control and analysis algorithms with realistic rigid body physics simulation.

For traditional joint-actuated robots, Gazebo [19] has been a popular physics simulation environment with a growing community, where common robots such as the PR2 [20] and TurtleBot [21], can be freely accessed and conveniently used directly *out of the box*. Furthermore, Gazebo uses the Robot Operation System (*ROS*) [22] such that it is convenient to incorporate different custom developed controllers and analysis modules. Moreover, simulations performed in Gazebo and ROS can then be seamlessly adapted to operate the corresponding robot hardware. However, Gazebo and ROS platforms do not directly support musculoskeletal robots.

In the field of biomechanics, OpenSim [23] allows the kinematics and dynamics of musculoskeletal systems with physiological muscle models to be simulated. Although widely used within biomechanics, the software cannot be used to study and control musculoskeletal robots. For musculoskeletal robots, an open-source simulation and analysis software *CASPR* [24] was developed. CASPR allows different cable-driven robot models, analysis algorithms and controllers to be conveniently simulated. In addition, CASPR could also be used to operate real cable-driven robot systems in a seamless manner [11], [25].

However, there are several limitations in CASPR within the study of musculoskeletal robots. First, the specification of robot, such as rigid body and cable attachment properties, using XML files is inconvenient and prone to errors. Second, the physics simulation used within CASPR is very simple and does not support features such as collision and contacts. Finally, the visualization capabilities in CASPR are very limited and do not allow a multiple object environment to be constructed. To the best of the authors' knowledge, there does not exist any complete simulation platform for musculoskeletal robots that combines the design with realistic physics simulation, while allowing the incorporation of muscle models and motion controllers.

In this paper, an open-source end-to-end design and physics simulation framework (named *CARDSFlow*) for generalized musculoskeletal robots is presented. The framework consists of three modules that connects a computer-aided design (*CAD*) software (*Fusion 360*), the Gazebo simulator and CASPR into one single framework. This allows: 1) 3D CAD of musculoskeletal robots to be automatically converted for use with Gazebo and CASPR; 2) physics simulation of musculoskeletal robots within Gazebo; and 3) integration of CASPR with CARDSFlow through the ROS platform. This framework allows a convenient way to design, simulate and control musculoskeletal robots. Furthermore, two different controllers, one in joint space and one in operational space, specific to musculoskeletal robots have been implemented in CARDSFlow. Through two examples, a two-link planar robot and the Roboy robot arm, the simplicity and practicality of CARDSFlow to simulate and operate physical musculoskeletal robots are demonstrated.

II. Framework of CARDSFlow

CARDSFlow consists of three components:

1) Fusion 360 CAD software to design the musculoskeletal robot (Section III). A module (*SDFusion*) is developed such that 3D CAD models of musculoskeletal robots can be automatically converted into SDF and XML files that are compatible with Gazebo and CASPR, respectively.
2) Gazebo physics simulator with support for musculoskeletal robots (Section IV). This is achieved through the *Muscle Model Module* developed for Gazebo that determines the muscle forces acting on the robot.

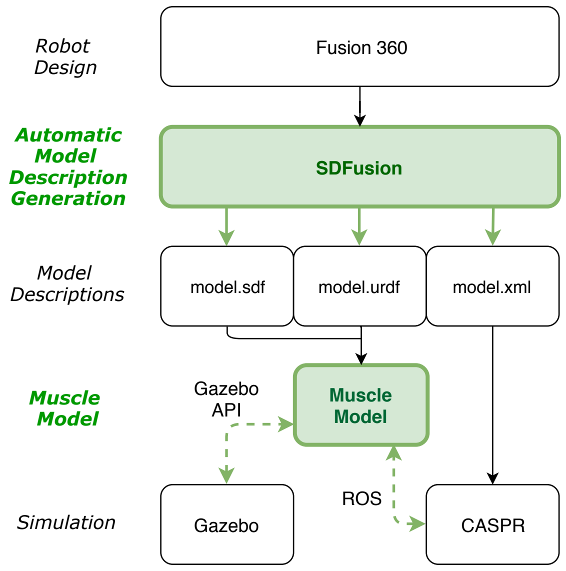


Fig. 2: CARDSFlow pipeline

3) Development of controllers and analysis algorithms through CASPR (Section V). This is achieved through the ROS framework to communicate between CASPR and Gazebo.

Figure 2 shows the flow, components and their interactions in the CARDSFlow system. The key characteristic and advantage of CARDSFlow is that the framework can be considered as end-to-end, where from the CAD software design of the robot to the control within a well-accepted physics environment allow musculoskeletal robots to be conveniently studied (Figure 3). Furthermore, by using the same ROS topics, controllers tested in physics simulations can be seamlessly adapted to the control of hardware robot systems [11]. As such, the use of Fusion 360, Gazebo and CASPR brings the benefits of the existing software together, as it will be described in the following sections.

III. From the Computer Aided Designs to Gazebo

For the accurate simulation of robots, an accurate model of the system is extremely important. Furthermore, the model description is also required within model-based controllers for the feed-forward control command. For musculoskeletal robots, the model information, including the mass, inertia, and muscle attachment points (or *via* points), is used in both the physics simulation (Gazebo) and controller design (CASPR). Within Gazebo and CASPR, user-defined SDF and XML files are typically hand-crafted manually, which is inconvenient and prone to errors.

Designing robots with 3D computer-aided design (*CAD*) software is therefore advantageous, as model descriptions can be easily obtained from the constructed CAD robot model. Using Fusion 360 from Autodesk, different parts of musculoskeletal robots can be easily designed (Figures 4a-4d). However, the existing version of Fusion 360 does not support the definition of the locations of via points, which are critical in the modelling of musculoskeletal robots.

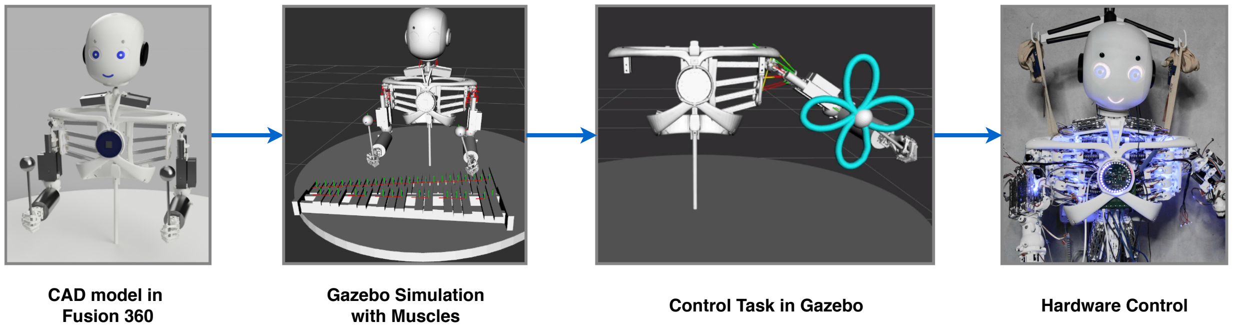
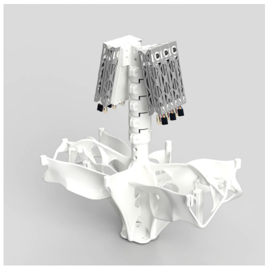


Fig. 3: CARDSFlow pipeline showing the flow: 1) designing the robot in Fusion 360; 2) simulation environment in Gazebo; 3) muscle control using CASPR in Gazebo; and 4) control of corresponding robot hardware.

a) Head and neck

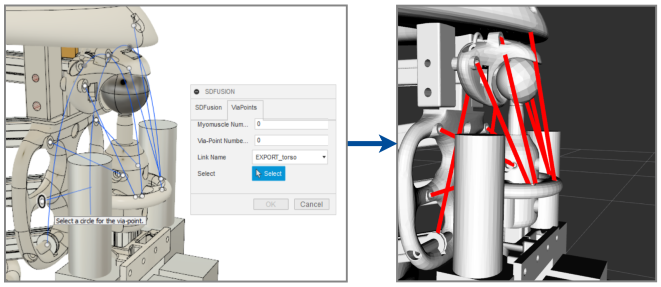
b) Left arm



c) Hip and Spine

d) Lower body

Fig. 4: Roboy 2.0 models created in Fusion 360



SDFUSION

SDFusion | ViaPoints

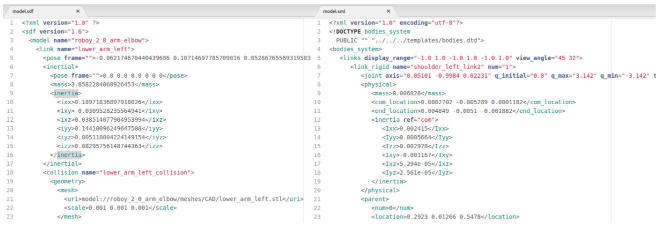
Myomuscle Num... 0

Via-Point Numbe... 0

Link Name EXPORT_torso

Select

Fig. 5: Create muscle models by defining via points. Left: Fusion 360 interface. Right: Resulting Gazebo simulation.



model.sdf

```
<?xml version="1.0" ?>
<sdf version="1.6">
  <model name="roboy_2_0_arm_elbow">
    <link name="lower_arm_left">
      <pose frame="">-0.062174670440439686 0.10714697785709816 0.05286765569319583
      <inertial>
        <pose frame="">0.0 0.0 0.0 0 0 0</pose>
        <mass>3.8582284060926453</mass>
        <inertia>
          <ixx>0.18971836897918026</ixx>
          <ixy>-0.0309528235564941</ixy>
          <ixz>0.038514077904953994</ixz>
          <iyy>0.14410096249647508</iyy>
          <iyz>0.005118004224149154</iyz>
          <izz>0.08295756148744363</izz>
        </inertia>
      </inertial>
      <collision name="lower_arm_left_collision">
        <geometry>
          <mesh>
            <uri>model://roboy_2_0_arm_elbow/meshes/CAD/lower_arm_left.stl</uri>
            <scale>0.001 0.001 0.001</scale>
          </mesh>
```

model.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE bodies_system
  PUBLIC "" "../../../templates/bodies.dtd">
<bodies_system>
   <links display_range="-1.0 1.0 -1.0 1.0 -1.0 1.0" view_angle="45 32">
      <link_rigid name="shoulder_left_link2" num="1">
         <joint axis="0.05101 -0.9984 0.02231" q_initial="0.0" q_max="3.142" q_min="-3.142" t
         <physical>
            <mass>0.006828</mass>
            <com_location>0.0002702 -0.005289 0.0001182</com_location>
            <end_location>0.004849 -0.0051 -0.001882</end_location>
            <inertia ref="com">
               <Ixx>0.002415</Ixx>
               <Iyy>0.0005664</Iyy>
               <Izz>0.002978</Izz>
               <Ixy>-0.001167</Ixy>
               <Ixz>5.294e-05</Ixz>
               <Iyz>2.561e-05</Iyz>
            </inertia>
         </physical>
         <parent>
            <num>0</num>
            <location>0.2923 0.01266 0.5478</location>
```

Fig. 6: Model Description Files exported using SDFusion (SDF, XML)

To extend the capability of Fusion 360 and support the design of via points, a Python module (*SDFusion*) is developed using Fusion's Python API. Users can define via points for the robot through the GUI, and the defined model can be automatically exported into SDF and XML model description files that are required by Gazebo and CASPR, respectively.

A. Definition of Via Points

Muscles can be modelled by defining the via points, which can be intuitively performed using the GUI of SDFusion in Fusion 360 (Figure 5). Users only need to construct a point on the model and select the link on which the muscle is attached. For each muscle model, multiple via points can be set to create multiple muscle segments.

B. Conversion to Model Description Files

SDFusion extracts link and joint information from the CAD model and converts the kinematic model into tree-structured model descriptions in SDF and XML formats (Figure 6). Model properties such as the center of mass and inertia of each link are calculated automatically by defining the material properties. User-defined via points are saved in both SDF and XML files, which are parsed by the Muscle Model Module (Section IV) and CASPR, respectively.

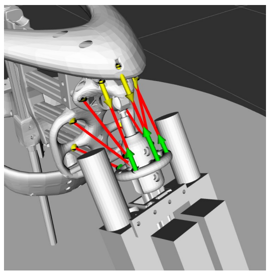
IV. Physics Modelling and Gazebo

A. General model of musculoskeletal robots

Musculoskeletal robots can be modelled as a multi-link cable-driven robot (*MCDR*), as proposed in [13]. The system dynamics of an n-DoF (*degree of freedom*) musculoskeletal robots with m muscles can be described by the following equation of motion (*EoM*):

$$M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) + \mathbf{G}(\mathbf{q}) + \mathbf{w}_e = -L(\mathbf{q})^T \mathbf{f}, \qquad (1)$$

where $\mathbf{q} \in \mathbb{R}^n$ is the generalized coordinates (*pose*), $M(\mathbf{q}) \in \mathbb{R}^{n \times n}$ is the mass-inertia matrix, $\mathbf{C}(\dot{\mathbf{q}}, \mathbf{q}) \in \mathbb{R}^n$ is the centrifugal and Coriolis vector, $\mathbf{G}(\mathbf{q}) \in \mathbb{R}^n$ is the gravitational vector, and $\mathbf{w}_e \in \mathbb{R}^n$ is the external wrench applying on the system. The joint-muscle Jacobian matrix $L(\mathbf{q}) \in \mathbb{R}^{m \times n}$


a) Muscle force visualization

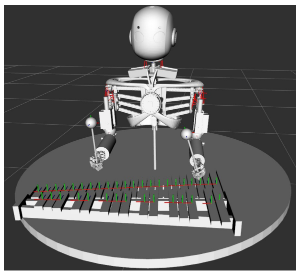

b) Roboy model with xylophone (Coordinate frames shown)

Fig. 7: Simulation environment in RViz

relates the muscle force vector $\mathbf{f} \in \mathbb{R}^m$ to the system wrench. Muscle forces are bounded by a set of positive minimum and maximum feasible muscle forces, f_{min} and f_{max}, such that

$$0 \leq f_{min} \leq f_i \leq f_{max}, \quad \forall i : f_i \in \mathbf{f} \tag{2}$$

B. Gazebo Muscle Model Module

With the SDF model descriptions, Gazebo constructs the robot model and generates the rigid body dynamics terms (left-hand side of (1)). However, the calculation of the joint-muscle Jacobian matrix $L(\mathbf{q})$ is not supported, such that the simulation of cable-driven robots is not directly available.

To extend Gazebo's capability to support cable-driven robots, the *Muscle Model Module* is developed using the extendable feature of SDF files and Gazebo C++ API. The via points defined in SDFusion are saved in the same SDF file with other kinematic properties, and are parsed to generate muscle routings. Furthermore, this module allows complex muscle kinematics, such as the wrapping of muscles over bone structures, to be modelled and passed into Gazebo.

Within Gazebo, a pair of tension forces are applied for each muscle segment at the via points as specified by the SDF file. This allows the effect of the muscle forces acting on the robot to be simulated in the Gazebo physics. Moreover, the module allows different muscle actuator models, such as cables, pneumatic air muscles, or even physiological muscles and ligaments, to be easily simulated.

The kinematics and dynamics of muscles computed by the module allow Gazebo to realistically simulate the physics of musculoskeletal robots. Additionally, RViz allows the results of the simulation to be visualized (Figure 7a), where the muscle forces are shown as arrows. Additional information about the model, such as coordinate frames and joint angles, can also be shown in RViz to monitor the robot's state.

The primary advantages of using Gazebo within the CARDSFlow framework include: 1) leverages a well-accepted realistic physics simulation platform using engines such as Bullet and ODE; 2) allows multiple robots and object environments to be easily simulated (Figure 7b); and 3) considers interactions between the robot and environment such that contacts can be simulated and controlled.

V. Integration of CASPR Controllers

A. CASPR controllers

In this section, two controllers for musculoskeletal robots, in the joint and operational spaces, have been implemented in CASPR as part of the CARDSFlow framework.

1) Joint Space CTC: The joint space computed-torque based controller (*CTC*) proposed in [26] is implemented to solve the joint space tracking problem. Given desired joint position $\mathbf{q}$, velocity $\dot{\mathbf{q}}$ and acceleration $\ddot{\mathbf{q}}$ trajectories, a set of force commands $\mathbf{f}$ is determined to actuate the system by solving the following optimization problem:

$$\begin{aligned} \underset{\mathbf{f}}{\text{minimize}} \quad & \frac{1}{2}\mathbf{f}^{\mathrm{T}} H_f \mathbf{f} \\ \text{subject to} \quad & M(\mathbf{q})\,\mathbf{a_c} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = -L^{\mathrm{T}}(\mathbf{q})\,\mathbf{f} \\ & \mathbf{a_c} = \ddot{\mathbf{q}}_\mathbf{d} + K_d(\dot{\mathbf{q}}_\mathbf{d} - \dot{\mathbf{q}}) + K_p(\mathbf{q_d} - \mathbf{q}) \\ & \mathbf{0} \leq \mathbf{f_{min}} \leq \mathbf{f} \leq \mathbf{f_{max}}, \end{aligned} \tag{3}$$

where the cable forces are minimized with the weight H_f and the joint position and velocity errors are compensated by the positive definite gains K_p, K_d. The force solution are bounded by the positive force constraints $\mathbf{f_{min}}$ and $\mathbf{f_{max}}$.

2) Operational Space CTC: In addition to the joint space, tracking tasks can be defined in operational space, for example, the position of the hand rather than the joint angles of the shoulder and elbow. To resolve the redundancy due to the higher DoF in joint space compared with the operational space, the following formulation is proposed to obtain the muscle forces $\mathbf{f}$ given operational space trajectories:

$$\begin{aligned} \underset{\ddot{\mathbf{q}}, \mathbf{f}}{\text{minimize}} \quad & \frac{1}{2}\ddot{\mathbf{q}}^T H_q \ddot{\mathbf{q}} + \frac{1}{2}\mathbf{f}^T H_f \mathbf{f}, \\ \text{subject to} \quad & M(\mathbf{q})\,\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = -L^{\mathrm{T}}(\mathbf{q})\,\mathbf{f} \\ & (\ddot{\mathbf{y}}_\mathbf{d} - \ddot{\mathbf{y}}) + K_d(\dot{\mathbf{y}}_\mathbf{d} - \dot{\mathbf{y}}) + K_p(\mathbf{y_d} - \mathbf{y}) = \mathbf{0} \\ & \dot{\mathbf{y}} = J(\mathbf{q})\dot{\mathbf{q}} \\ & \ddot{\mathbf{y}} = J(\mathbf{q})\ddot{\mathbf{q}} + \dot{J}(\mathbf{q})\dot{\mathbf{q}} \\ & \mathbf{0} \leq \mathbf{f_{min}} \leq \mathbf{f} \leq \mathbf{f_{max}}, \end{aligned} \tag{4}$$

where the cost function consists of joint acceleration $\ddot{\mathbf{q}}$ and cable force $\mathbf{f}$. The vector $\mathbf{y}$ refers to the operational space position, $J(\mathbf{q})$ refers to the joint-to-operational space Jacobian, and the error dynamics in operational space is controlled by the positive definite gains K_p and K_d.

B. Integration of CASPR controllers and Gazebo

Communication between CASPR controllers and Gazebo is established through the Robotics System Toolbox in MATLAB and the Muscle Model Module. CASPR achieves ROS communication through the Robotics System Toolbox, and the Muscle Model Module communicates with the CASPR controller with ROS messages and the Gazebo simulator with the Gazebo API (Figure 8).

Given joint position $\mathbf{q}$ and velocity $\dot{\mathbf{q}}$, force commands $\mathbf{f}$ generated by the CASPR controllers are sent to the Muscle Model Module through ROS messages. The module then applies the set of forces on the Gazebo models. Feedback $\mathbf{q}$ and $\dot{\mathbf{q}}$ are published to the ROS topic that CASPR

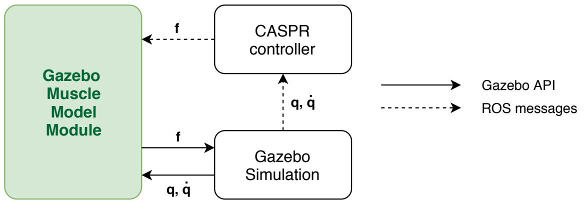


Fig. 8: Communication between CASPR controllers and Gazebo

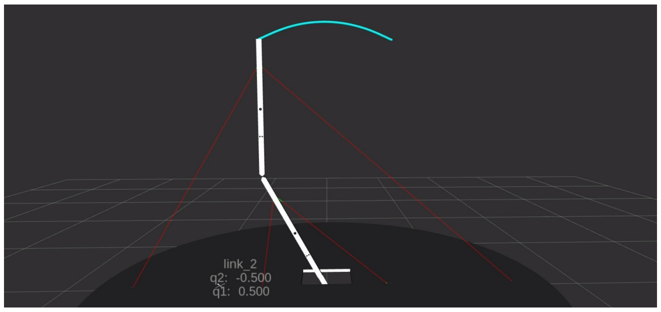
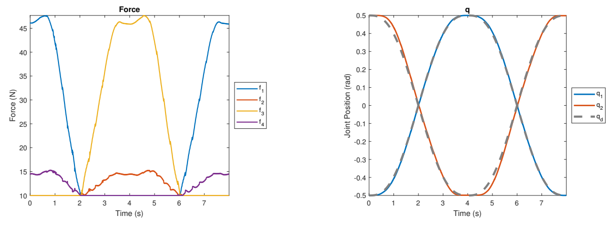


Fig. 9: 2R-Planar robot tracking a joint space trajectory in Gazebo simulation



a) Force commands $\mathbf{f}$ b) Joint Position $\mathbf{q}$

Fig. 10: Joint Space Control on 2R-Planar robot

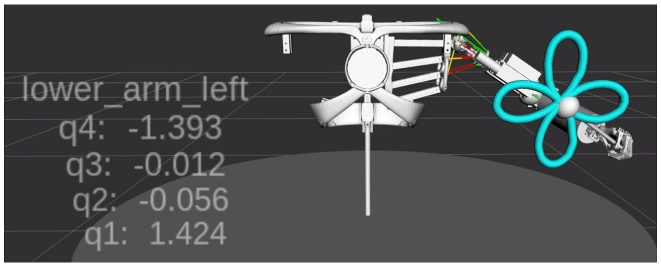


Fig. 11: Roboy's left arm tracking a flower-like trajectory in Gazebo simulation

has subscribed, such that the control loop is established. Furthermore, by using the same ROS topics as in simulations, the controllers tested in Gazebo can be seamlessly adapted to the control of hardware robot systems [11].

VI. Simulation Examples

In this section, simulation examples of controlling two musculoskeletal robots, 2R-Planar robot and Roboy's left arm, are presented to demonstrate the capability of CARDS-Flow. Joint space control on the 2R-Planar robot is first presented, followed by the operational space control on Roboy's left arm. Kinematics and control inputs are saved and analyzed in CASPR, with results shown in figures 9-12.

Simulations are conducted on two sets of hardware connected through local network. The first set contributes as the platform for the Gazebo simulation environment, with ROS-Kinetic running on Ubuntu 16.04 LTS. The second set of hardware runs CASPR on MATLAB R2018a, Windows 8.1. This setup shows that multiple machines and operating systems can be involved in the framework of CARDSFlow.

A. 2R-Planar Robot

The 2R-Planar robot is a 2-link, 2-DoF planar robot with 2 revolute joints (Figure 9). The 2 links are actuated by 4 muscles, which are visualized with red lines. As the end-effector, the path of the second link's tip is represented by a series of light blue spheres. Joint positions $[\mathbf{q_1},\mathbf{q_2}]$ are displayed to keep track of the robot's status.

Using the joint space CTC in CASPR, a joint space trajectory is tracked. The trajectory is defined between $\mathbf{q} = [-0.5, 0.5]$, $[0.5, -0.5]$, $[-0.5, 0.5]$, and lasts for 8 seconds. The controller gains are set to $K_p = 2000$ and $K_d = 90$ (damping ratio ≈ 1). With the force commands shown in figure 10a, it can be seen from figure 10b that the controller is capable of tracking the reference trajectory (dotted lines).

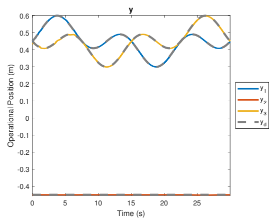
B. Roboy's Left Arm

The left arm of Roboy is a 2-link 4-DoF musculoskeletal structure, with 3-DoF at the shoulder joint and 1-DoF at the elbow joint (Figure 11). The arm is actuated by 11 cables, with 9 actuating the upper arm, and 2 actuating the lower arm. The end-effector is defined on the tip of the stick that the arm is holding. Similarly, a series of light blue spheres is rendered to indicate the end-effector's path.

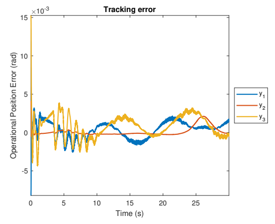
The robot is required to track a flower-like trajectory on the plane $\mathbf{y_2} = -0.45$. The flower trajectory has a center at $\mathbf{y} = [0.45, -0.45, 0.45]$, an amplitude of 0.15 and lasts for 30 seconds. The light blue flower shown in figure 11 is the actual path of the end-effector. Figure 12 shows that the operational space CTC is capable of controlling the robot's end-effector to track the reference trajectory. This result verifies the operational space CTC formulation on a complex structure, and demonstrates the convenience of incorporating custom developed controllers into CARDSFlow.

VII. Conclusion and Future Works

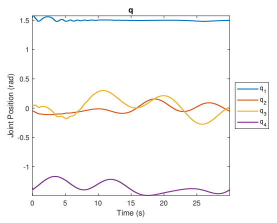
An end-to-end physics simulation framework that connects computer-aided design, modelling, to simulation and control of musculoskeletal robots was presented. Automatic and convenient conversion from CAD models to model description files compatible with Gazebo and CASPR was made possible with SDFusion. Using the Muscle Model Module, the modelling and visualization of musculoskeletal robots in the Gazebo environment with a range of useful features was achieved. The module also facilitates the communication between CASPR controllers and the Gazebo simulation. Two control tasks in joint space and operational space were



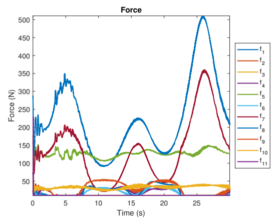
a) Operational Position $\mathbf{y}$



b) Operational Position Error $\mathbf{e_y}$



c) Joint Position $\mathbf{q}$



d) Force commands $\mathbf{f}$

Fig. 12: Simulation Results of Operational Space Control on Roboy's left arm

performed and the results demonstrated the practicality of CARDSFlow in the operation and controller design of musculoskeletal robots. The variety in the demonstrated robots also showed the potential of extending the framework to other classes of musculoskeletal robots.

Acknowledgement

The work was supported by the European Unions Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreement No. 785907 (Human Brain Project SGA2), the Germany/Hong Kong Joint Research Scheme sponsored by the Research Grants Council of Hong Kong and the German Academic Exchange Service of Germany (Reference No. G-CUHK410/16), and the Early Career Scheme sponsored by the Research Grants Council (Reference No. 24200516).

References

[1] I. Kato, S. Ohteru, K. Shirai, T. Matsushima, S. Narita, S. Sugano, T. Kobayashi, and E. Fujisawa, "The robot musician 'wabot-2' (waseda robot-2)," *Robot.*, vol. 3, no. 2, pp. 143–155, 1987.

[2] A. Ramezani, S.-J. Chung, and S. Hutchinson, "A biomimetic robotic platform to study flight specializations of bats," *Sci. Robot.*, vol. 2, p. eaal2505, 2017.

[3] Z. G. Zhang, N. Yamashita, M. Gondo, A. Yamamoto, and T. Higuchi, "Electrostatically actuated robotic fish: Design and control for high-mobility open-loop swimming," *IEEE Trans. Robot.*, vol. 24, pp. 118–129, 2008.

[4] N. Perrin, D. Lau, and V. Padois, "Effective generation of dynamically balanced locomotion with multiple non-coplanar contacts," in *Robotics Research*, ser. Springer Proceedings in Advanced Robotics, A. Bicchi and W. Burgard, Eds. Springer, Cham, 2018, vol. 3, pp. 201–216.

[5] N. G. Tsagarakis, G. Metta, G. Sandini, D. Vernon, R. Beira, F. Becchi, L. Righetti, J. Santos-Victor, A. J. Ijspeert, M. C. Carrozza, and D. G. Caldwell, "icub: the design and realization of an open humanoid platform for cognitive and neuroscience research," *Adv. Robot.*, vol. 21, no. 10, pp. 1151–1175, 2007.

[6] A. Pott, H. Mütherich, W. Kraus, V. Schmidt, P. Miermeister, and A. Verl, "IPAnema: A family of cable-driven parallel robots for industrial applications," in *Cable-Driven Parallel Robots*, ser. Mechanisms and Machine Science, A. Pott and T. Bruckmann, Eds. Springer International Publishing, 2013, vol. 12, pp. 119–134.

[7] S. Wittmeier, C. Alessandro, N. Bascarevic, K. Dalamagkidis, D. Devereux, A. Diamond, M. Jäntsch, K. Jovanovic, R. Knight, H. G. Marques, P. Milosavljevic, B. Mitra, B. Svetozarevic, V. Potkonjak, R. Pfeifer, A. Knoll, and O. Holland, "Toward anthropomimetic robotics: Development, simulation, and control of a musculoskeletal torso," *J. Artif. Life*, vol. 19, no. 1, pp. 171–193, 2013.

[8] B.-S. Kang, C. S. Kothera, B. K. S. Woods, and N. M. Wereley, "Dynamic modeling of mckibben pneumatic artificial muscles for antagonistic actuation," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 182–187.

[9] B. Verrelst, R. V. Ham, B. Vanderborght, F. Daerden, D. Lefeber, and J. Vermeulen, "The pneumatic biped "lucy" actuated with pleated pneumatic artificial muscles," *Auton. Robot.*, vol. 18, no. 2, pp. 201–213, 2005.

[10] R. Pfeifer, P. Y. Tao, H. G. Marques, S. Weydert, D. Brum, M. Weyland, R. Hostettler, F. Volkert, V. Gmünder, and D. Halbeisen. Roboy anthropomimetic robot. [Online]. Available: www.roboy.org

[11] J. Eden, C. Song, Y. Tan, D. Oetomo, and D. Lau, "CASPR-ROS: A generalised cable robot software in ROS for hardware," in *Cable-Driven Parallel Robots*. Springer International Publishing, 2018, pp. 50–61.

[12] Y. Asano, S. Nakashima, T. Kozuki, S. Ookubo, I. Yanokura, Y. Kakiuchi, and M. I. Kei Okada, "Human mimetic foot structure with multi-DOFs and multi-sensors for musculoskeletal humanoid kengoro," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 2419–2424.

[13] D. Lau, D. Oetomo, and S. K. Halgamuge, "Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix," *IEEE Trans. Robot.*, vol. 29, no. 5, pp. 1102–1113, 2013.

[14] D. Lau, J. Eden, D. Oetomo, and S. K. Halgamuge, "Musculoskeletal static workspace of the human shoulder as a cable-driven robot," *IEEE/ASME Trans. Mechatronics*, vol. 20, no. 2, pp. 978–984, 2015.

[15] D. Lau, J. Eden, S. Halgamuge, and D. Oetomo, "Cable function analysis for the musculoskeletal static workspace of a human shoulder," in *Cable-Driven Parallel Robots*, ser. Mechanisms and Machine Science, A. Pott and T. Bruckmann, Eds. Springer International Publishing, 2015, vol. 32, pp. 263–274.

[16] D. Lau, D. Oetomo, and S. K. Halgamuge, "Inverse dynamics of multilink cable-driven manipulators with the consideration of joint interaction forces and moments," *IEEE Trans. Robot.*, vol. 31, no. 2, pp. 479–488, 2015.

[17] M. Jäntsch, C. Schmaler, S. Wittmeier, K. Dalamagkidis, and A. Knoll, "A scalable joint-space controller for musculoskeletal robots with spherical joints," in *IEEE Int. Conf. Robot. and Biomimetics*, 2011, pp. 2211–2216.

[18] J. Eden, Y. Tan, D. Lau, and D. Oetomo, "Reference state trajectory generation for output tracking with constraints using search trees," in *American Contr. Conf.*, 2018, pp. 4682–4687.

[19] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2004, pp. 2149–2154.

[20] S. Cousins, "ROS on the PR2 [ROS topics]," *IEEE Robot. Autom. Mag.*, vol. 17, no. 3, pp. 23–25, 2010.

[21] W. Garage. Turtlebot. [Online]. Available: https://www.turtlebot.com/

[22] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Ng, "ROS: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[23] 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 Trans. Biomed. Eng.*, vol. 54, no. 11, pp. 1940–1950, 2007.

[24] D. Lau, J. Eden, Y. Tan, and D. Oetomo, "CASPR: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2016, pp. 3004–3011.

[25] Y. Wu, H. H. Cheng, A. Fingrut, K. Crolla, Y. Yam, and D. Lau, "CU-brick cable-driven robot for automated construction of complex brick structures: From simulation to hardware realisation," in *IEEE Int. Conf. Sim. Model. Program. Auton. Robot.*, 2018, pp. 166–173.

[26] A. B. Alp and S. K. Agrawal, "Cable suspended robots: Design, planning and control," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2002, pp. 4275–4280.