A Generic Simulator for Underactuated Compliant Hands

Alessio Rocchi[1] and Kris Hauser[2]

***Abstract*— Compliant underactuated hands have been shown to be able to grasp a variety of objects while simplifying both mechanical and control complexity compared to fully-actuated hands. Building on recent advances in simulation software, this paper presents a generic underactuated compliant hand emulator working with state of the art simulation software that allows to evaluate the behavior of a compliant gripper grasping irregular objects. The emulator makes use of the adaptive synergy concept which can be generalizable to a wide array of underactuated compliant hands. The architecture of the simulator-emulator system is presented and the implementation details will be highlighted for two common compliant grippers, the Pisa/IIT SoftHand and the RightHand Robotics Reflex SF. These software tools are being used for the simulation track of the IROS 2016 Robot Grasping and Manipulation Competition in which teams are asked to develop novel controllers to grasp a variety of cluttered and irregular objects.**

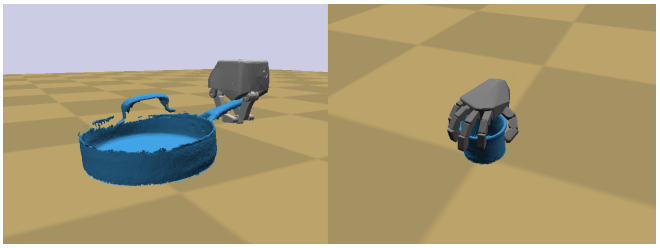

Fig. 1. Two hands simulated using the proposed emulator built atop the Klamp't simulator. On the left, the RightHand Robotics Reflex SF, on the right the Pisa/IIT SoftHand grasping objects from the *YCB* dataset, one of the test sets available in the simulation framework for the virtual track of the IROS 2016 Robotic Grasping and Manipulation Competition, available at `git.io/viere`

I. Introduction

Compliant underactuated hands (CUHs) have become popular in robotics because of their adaptability, robustness and forgiveness to sensor uncertainties, with notable examples such as the iHY [1], Reflex Hand [2], Pisa/IIT SoftHand [3], Robotiq 3-Fingered Gripper [4], RBO [5] and RBO2 [6] and Yale Hand [7], and with interesting examples of applications in prosthetics [8].

Underactuated hands provide many degrees of freedom (DoFs) while being actuated by a lower number of degrees of actuation (DoAs). Compliant mechanisms help improve hardware robustness because they can withstand accidental contact with the environment. At the same time, passive mechanisms allow the hand shape to adapt to a wide range of grasped objects without the active control of actuators via tactile sensors, and hence underactuation theoretically simplifies the control problem by delegating to the hardware design a degree of adaptivity. However, the benefits of compliance and underactuation typically come at a price, because they introduce sensing and actuation uncertainties which are largely intrinsic to the design. For example, it may not be possible to extend or contract a single joint independently from the rest of a phalange, or to apply stiffer forces in a given direction while keeping the configuration fixed.

Model-based predictions of motions and forces can offer powerful insights for grasp planning [9], [10], and are an invaluable tool in testing high level grasp controllers before application to physical hardware. Simulation tools will help predict outcomes of controllers that make deliberate contact with the environment [11] as well as with the object. They may also be useful for computational exploration of gripper designs as well as control strategies, such as in reinforcement learning.

Despite increasing interest in CUHs, to the knowledge of the authors no tool has been presented that allows for the dynamic simulation of generic compliant mechanisms. This work has been enabled by recent advances in simulation software [12] and stems from the observation that a wide range of hands can be modeled using a framework that extends the concept of soft synergies into that of adaptive synergies [13] and can be useful to model CUHs with a known transmission distribution matrix and linear joint stiffness.

We present a generic compliant underactuated hand emulator that is built upon the open source Klamp't simulator (`http://klampt.org`) and provides a common tool for simulating a large set of CUHs. The underlying simulator allows to simulate interaction with rigid objects taking into account the full dynamics of the hand, object and environment. It integrates models of compliant joints with recent contributions in robust mesh-mesh contact generation methods [14], and uses adaptive time stepping capabilities to reduce penetration artifacts when simulating contact compliance. The presented emulator design allows to simulate a new CUH writing a minimal amount of code. The architecture of the emulator-simulation system will be described and details on the implementation of two popular CUHs will be given. Finally, this paper will describe how the simulator is being used in the IROS Robotic Grasping and Manipulation Competition, where a virtual track has been setup to allow participants to compete in a simulated grasping competition which will take place alongside the physical competition in October 2016. The novel work in this paper provides a hand emulator to help teams easily incorporate a CUH as a simulated gripper in the competition's virtual track.

[1]Alessio Rocchi is with the Department of Advanced Robotics, Istituto Italiano di Tecnologia, Via Morego 30, 16163, Genova, Italy `alessio.rocchi@iit.it`

[2]Kris Hauser is with the Department of Electrical & Computer Engineering, Duke University, Durham, NC 27708, USA `kris.hauser@duke.edu`

II. Related Work

Gazebo [15] is one of the most popular general-purpose robot simulators, and is built with flexibility in mind: it supports a very versatile and expressive Simulation Description Format (SDF), through which a variety of mechanisms can be modeled and simulated, including grippers. The simulator allows to pick between several physics engines in order to perform the actual physical simulation, and contact stiffness (but not joint stiffness) are supported both at the level of the description format and of the simulator. Moreover a series of transmission mechanisms is implemented out of the box, with work underway to integrate *mimic* joint transmission, which will allow to describe simple joint coupling and implement them as constraints in the underlying physics engine solver. Moreover, Gazebo plugins have implemented physics engines with the recent boundary-layer expanded meshes (BLEM) [14] technique for stable contact generation [12]. Although there is existing work for simulating the Pisa/IIT SoftHand in Gazebo [16], there are no generic plugins which allow to simulate different CUHs flexibly at the moment.

A few robot simulators are specialized in grasping simulation and analysis, such as GraspIt! [17] and OpenGRASP [18] (built on top of the popular OpenRave [19] simulation framework) but assume fully actuated grippers and are not yet able to simulate a variety of CUHs.

In [20] a MATLAB toolbox for grasp analysis of CUHs is presented: compliance can be modeled at contact points, at the joints and on the actuation system (including transmission). The toolbox contains several functions for grasp analysis and mechanism optimization, it allows to easily model novel mechanisms and import existing designs, but lacks dynamic simulation capabilities.

In [21] a dynamic simulation of the Pisa/IIT SoftHand is implemented in the multi-body dynamics simulator MSC Adams [22]. The simulator is used to validate the provided methods to generate pre-grasp palm configurations w.r.t. the object pose [23]. The simulator demonstrated moderate fidelity to an experimental scenario, but the authors acknowledge difficulties with hand-object penetrations and estimation of contact normals.

The Klamp't simulator has been demonstrated to allow for stable simulation of CUHs [12] using a combination of three techniques boundary-layer expanded meshes (BLEM) [14], contact point clustering and adaptive time stepping. Moreover, recent developments of the simulator provide a uniform interface for specifying emulators for custom sensors and actuators/transmission systems, by using a flexible API and the Python language. These features enable the development of a lean architecture for developing a generic compliant hand emulator.

III. Compliant Hand Emulator

A. Summary

An underactuated hand is modeled as a set of rigid links articulated by n joints, with n_a degrees of actuation and $n_a < n$. The state of the fingers is denoted as $q \in \mathbb{R}^n$. A control $u \in \mathbb{R}^{n_a}$ gives rise to a net torque on the joints $\tau \equiv \tau(q,u) \in \mathbb{R}^n$ which summarizes the sum of internal torques including gearing, stiffness, damping, joint stops, and friction.

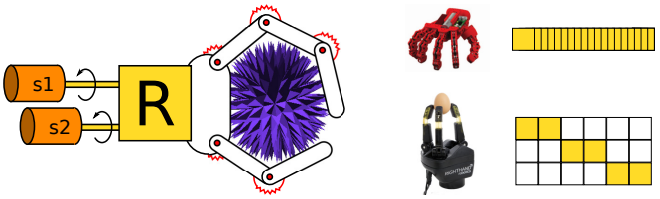

Fig. 2. On the left, a simple schematic showing a simplistic gripper with underactuated transmission, with 2 DoAs and 5 DoFs. On the right, the structure of the transmission distribution matrix for the SoftHand (top) and Reflex (bottom), with white cells where the value of the matrix is 0. For the Reflex, each row is obtained as a single Da Vinci mechanism with two phalanges, which can be derived from [24] by changing coordinates ($\theta_i = \frac{\pi}{2} - q_i$) and reordering T, obtaining for each finger $R_\mathrm{f} = [\,1 \; -R_1/r_0\,]$.

Thus, the dynamics of the robot in contact are given by

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = \tau(q,u) + \tau_c \qquad (1)$$

where $B(q)$ is the robot's mass matrix, $C(q,\dot{q})$ is the Coriolis force matrix, $G(q)$ is the generalized gravity vector, and $\tau_c = J^T f_c$ are the joint torques resulting from external contact forces.

Given an initial state $(q_0, \dot{q}_0)$, a control trajectory $u(t)$, and a final time T a simulator will generate a trajectory of the robot $q(t) : [0,T] \to \mathbb{R}^n$ as well as the motions of other objects $O_1, \ldots, O_m$, taking the dynamics into account as from (1).

A fundamental part of simulating grippers, and in particular CUHs lies in implementing a model of underactuation, to emulate the underactuated transmission system regardless on the particular model of hand and its kinematic and dynamic properties. The proposed emulator has the objective of providing such functionalities, and as described in section IV, it is loaded by Klamp't, together with the hand model providing its dynamic and kinematics properties, in order to dynamically simulate a CUH.

B. Modeling underactuation and compliance

Underactuated and compliant hands are linked with transmission mechanisms, e.g., tendons or mechanical linkages, that distribute actuator effort across multiple joints. They also include spring mechanisms that restore the hand to a consistent rest state once gripping effort is removed, that is, restore deterministic behavior to the hand which would be otherwise be caused by the underactuation (given a certain value for the actuated variables, an infinite number of possible configurations exist for the underactuated joints). Simulations must allow for actuators to drive multiple links forward, but also to allow for forces on one link to affect the distribution of effort across other links. We model these effects with a formulation based on the adaptive synergies framework [13].

First, we use a general constraint model that relates actuator displacements s to configuration displacements q:

$$s = Rq \qquad (2)$$

where the reference configuration is chosen so the zero actuator and joint correspond. The $n_A \times n$ transmission matrix R determines how joint movements pull on each actuator.

We assume the drive mechanism generates resultant torques on individual joints in order to maintain these constraints. Denote the tensional force at the tendons generated by the actuators as $f \in \mathbb{R}^n_a$, and the torques generated by the drive mechanism be denoted $\tau_d \in \mathbb{R}^n$. By the principle of virtual work, we have $\tau_d = R^T f$. Let us also define the joint torques produced by spring mechanisms as $\tau_s = -Eq$ where E is a $n \times n$ joint stiffness matrix (which is usually diagonal). Neglecting friction effects, the resultant vector of joint torques is

$$\tilde{\tau} = R^T f - Eq. \tag{3}$$

where $\tilde{\tau} = \tau_c$ at the equilibrium. We then solve for f and q that satisfy the constraints (2) and (3) by solving a system of linear equations, which has a closed form solution in terms of s, R, E, and τ [13] for a constant transmission distribution matrix R and joint stiffness matrix E :

$$f = AJ^T f_c + Bs \tag{4}$$

$$q = CJ^T f_c + Ds \tag{5}$$

with A, B, C, D properly defined matrices, as in [13]. Given the solved f from eq (4) and substituting the result in (3), we obtain joint torques to actuate on every joint given a position command s on actuators. Given (1) and imposing quasi-static equilibrium conditions, we then compute the gravity compensation terms and define the actuation torque as

$$\tau = \tilde{\tau} + G(q). \tag{6}$$

This procedure needs to iterate over multiple time steps to achieve equilibrium between mechanism torques and contact forces, which may cause chattering especially if contact forces are nonsmooth. At rest, the driven hand will converge to the configuration given by eq. (5), when the underactuated joints are modeled without friction. For hands with configuration dependent transmission distribution matrix R, we can still use the formula assuming small displacements in q imply little variations in R, and thus setting a number of simulation substeps $\gg 1$. Moreover, while equations (2)-(5) assume proper offsets in the actuator configuration and a zero rest position for the joint elastic elements, it will not be necessarily the case for the hand model, so that when emulating the underactuated transmission, we will resort to commanding a position command $\sigma \in [0, 1]$ such that

$$\sigma = \sigma_{\text{scaling}}(s - \sigma_{\text{offset}}) \tag{7}$$

and eq. (3) becomes

$$\tilde{\tau} = R^T f - E\,(q - q_{\text{rest}}) \tag{8}$$

Also, not necessarily all joints in the hand will be part of the underactuated transmission, so that all previous formulas do not apply to all joints, but to the subset of underactuated joints n_u. More details on this aspect will be given in the following section.

IV. Architecture

In Fig. 4 a typical architecture for a simulation program is shown. The user creates a program which may or may not (in case of "headless" simulation without a graphical interface) include a `GLSimulationProgram`, and which contains a `SimpleSimulator` instance, which extends the `Simulator` class with logging and mechanism emulation capabilities, and takes care to update the controllers and the sensors and actuators emulators attached to it at every simulation loop. While the `Simulator` class is accessed through an automatically generated Python binding of a C++ class compiled in Klamp't dynamic library, all other entities are implemented using Python 2.7.

Each outer simulation step may be composed of several substeps, during which the simulated world state evolves and sensors and actuators emulators are updated; the controller is updated once per every outer step while forces and dynamics are evolved at a higher rate. The user code takes care of loading and/or procedurally creating the environment and the robot, and sends commands to the robot via a high level controller. Zero or more sensor and actuator emulators can be attached to each controller. The standard sensors already implemented in Klamp't and included in the robot description will provide data that is then passed directly to the controller. The distinction between an emulator and controller is that the controller is a stand-in for a control system's software that repeatedly reads data from the sensors and output commands to actuators, while emulators have access to the whole "omniscient" state of the simulation and can apply forces at will.

A. *CompliantHandEmulator*

Compliant underactuated hands that can be modeled with adaptive synergies are implemented by subclassing the `CompliantHandEmulator` class. In Fig. 3 a simplified class diagram for the Reflex and Pisa/IIT SoftHand is shown. While the latter's transmission matrix is constant and depends on the ratios of the pulleys' radii, for the former it is configuration dependent and can be modeled as a Da Vinci mechanism [24]: for this reason the Reflex implements also the `updateR()` method. The major functions of the base `CompliantHandEmulator` are also shown in the figure.

To set parameters describing the structure of the hand, the subclass must implement the `loadHandParameters()` function. These parameters define the set of underactuated

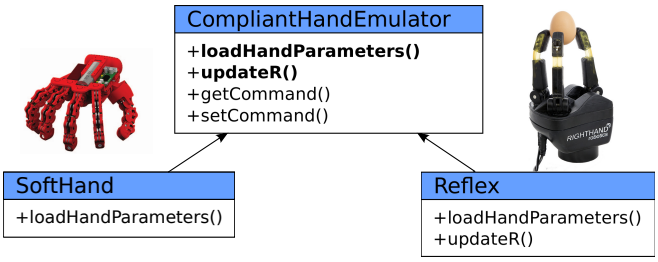

Fig. 3. Simplified class diagram scheme of the Reflex and Pisa/IIT SoftHand emulators.

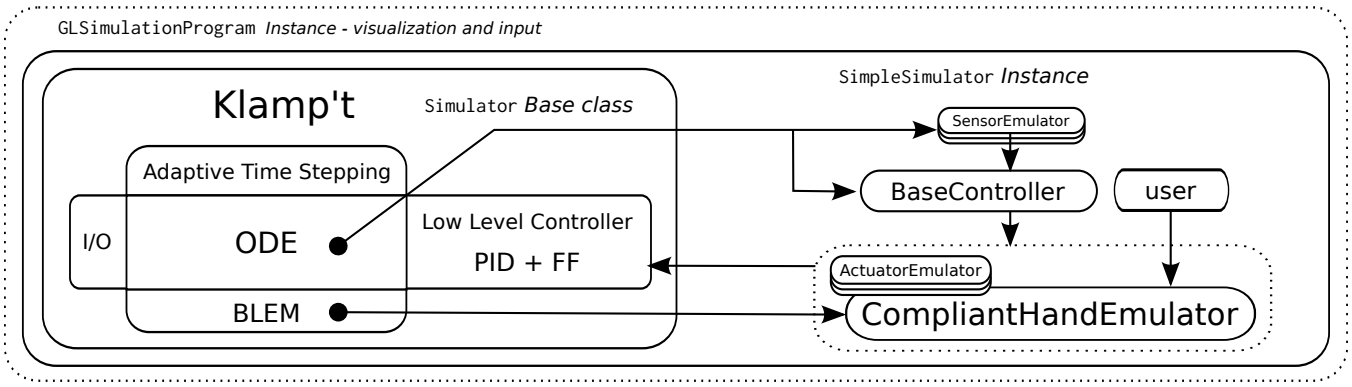

Fig. 4. While a `GLSimulationProgram` handles visualization and user input, a `SimpleSimulation` will instantiate a Klamp't simulator object, load a simulation environment (world), a robot and its controller. Each controller will receive simple input from the simulator, and from a series of `SensorEmulator` objects. Each one of them emulates an arbitrary sensory hardware. The control input is then passed to a series of `ActuatorEmulator` instances, where each one emulates a specific actuator/transmission type. The `CompliantHandEmulator` is an `ActuatorEmulator` which allows input either from a `BaseController` or directly from the user through the `getCommands()` and `setCommands()` calls. The user code will take care of creating the simulation scenario (through Klamp't I/O functionalities) and implement an high level controller (e.g. a state machine)

joints, the synergy joints (i.e. the joints for which a non-trivial transmission exists that relates it to the underactuated joints of the gripper), and the actuated joints with a standard transmission (i.e. joints with `Normal` or `Affine` transmission, which are the default transmission types implemented at the moment in Klamp't). This set of parameters is summarized in Table I, with full details found in the emulator documentation.

Fig. 4 also illustrates a few more details on the architecture of `CompliantHandEmulator`. Contact point and force information are obtained thanks to the BLEM [14] algorithm allowing for a stable emulation of the adaptive synergy [12]; thanks to adaptive time-stepping, mesh interpenetration are avoided even in the case of large contact forces where finite contact stiffness can cause the contact to happen below the mesh boundary layer. The relative smoothness of the BLEM contact force predictions are crucial to synergy emulation, since the external forces component f_c in (4) directly influence actuated torques.

The `CompliantHandEmulator` accepts commands both directly from the user (man-in-the-loop) or from a high level controller. It simulates simple joint transmission constraints (present for example in the SoftHand emulator due to the presence of roll-articular joints [3]) using a simple PID control with a setpoint updated to the reference (eventually underactuated) joint position at every simulation substep, while all the underactuated joints are commanded using eq. (4). The emulator also shows the predicted equilibrium configuration from eq. (5), with the predicted equilibrium configuration changing during the hand evolution because of the object dynamics during object-hand interaction.

For testing purposes, the emulator also provides an interface to easily define virtual contact and virtual forces in order to qualitatively inspect the adaptive behavior of the hand and troubleshoot issues on the transmission and stiffness matrix design.

Parameter	Meaning	Size
`u_to_n`	map of underactuated joint ids	n_u
`a_to_n`	maps joints ids to synergy actuators	n_a
`d_to_n`	maps joints ids to regular actuators	n_d
`E`	joint stiffness matrix	$n_u \times n_u$
`R`	transmission matrix	$n_a \times n_u$
`synergy_reduction`	scaling parameter σ_{scaling} in (7)	n_a
`q_u_rest`	underactuated joint spring rest position q_{rest} in eq. (8)	$0 \vee n_a$
`sigma_offset`	offset parameter σ_{offset} in eq. (7)	$0 \vee n_a$

TABLE I

Parameters typically set by the `loadHandParameters()` call in a new compliant hand emulator. Parameters that may be set to size 0 are optional (i.e. they are assumed to be zero-valued vectors of the correct size)

V. Experiments

As shown in Fig. 5, the simulator obtains a real time factor of ~ 0.5 with simple contact scenarios and ~ 0.24 in a typical grasp simulation with an i7-6500U @ 2.50GHz processor, 10 substeps (10ms per time step, 1ms every physics substep), with the adaptive timestep scheme requiring slower speeds for more complex contact scenarios or less stiff objects. In Fig. 6 the SoftHand grasps an object closing the hand in 0.5s and then lifting the object 0.1m with a simple linear trajectory of 1.5s. As depicted in the graph, the index finger's distal joint configuration converges to the computed equilibrium configuration, while the proximal joint does not reach equilibrium by the end of the simulation. An explanation can be understood by noticing the noisy value of the predicted equilibrium value which comes from noisy contact forces during the lift motion (e.g. τ_c proximal). It is also evident how the peaks in contact torques are related to the peaks in the tendon tension, and ultimately to the actuated joint torques of the emulated underactuated hand.

As shown in Fig. 3 emulators for both the Reflex Hand and the SoftHand are implemented by extending the `CompliantHandEmulator` class, in order to define the underactuated joints, the joint stiffness and the transmission

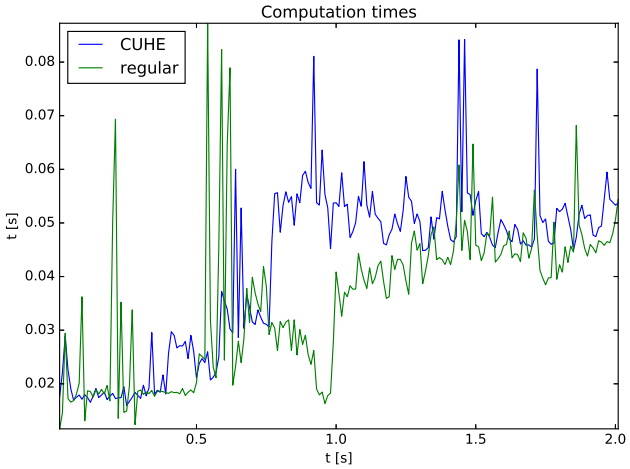

Fig. 5. Wall time for each simulation step (0.01s) during grasp of object `wearever_cooking_pan_with_lid` from the *YCB* set (Fig. 1) with CUHE and regular Reflex simulator.

distribution matrix. While the Reflex is loaded using the Klamp't native *ROB* format, for the SoftHand we use the *URDF* format from the hand, available in [16]. In particular, for the SoftHand, a `a SoftHandLoader` class has been created which is able to automatically load transmission and stiffness parameters form the URDF. The loader is thus usable as a reference for hands whose parameters are encoded in the model's URDF. Both hands are implemented by using virtual joints being used as synergistic actuators, in particular the joint driving `invisible_wire_link` in case of the SoftHand, and `wire_1`, `wire_2`, `wire_3` for the Reflex. This is a design choice which is enforced in the emulator code and should be kept in new hand models whose transmission is simulated through the emulator. For the three-fingered ReFlex, there are 4 DoAs, with 1 DoA and 2 DoFs for each finger plus an additional pregrasp mechanism which changes the direction in which finger 2 and 3 close from power grasp to pinch grasp. Assuming the distal joints rotate along a fixed axis, $n = 7$, but in general the soft joint between the proximal and distal joints may flex and stretch. The swivel joints in the Reflex are implemented using the `Affine` transmission in Klamp't, and thus the two joints are modeled as a single regular actuator in the `CompliantHandEmulator` ($n_d = 1$), with three more synergistic actuators, one for each finger ($n_a = 3$), distributing the actuator torque over two joints. For the five-fingered SoftHand, $n_a = 1$, $n_u = 19$, with one abduction/adduction joint for each finger, plus 3 DoFs except for the thumb (which has 2) which are rolling-contact joints [3]. Hence to the 19 underactuated joints correspond 14 additional joints, since each rolling-contact (Hillberry) joint is modeled using one underactuated joint and one matched joint constrained to follow its parent joint movements [21].

A. Use in a Simulation Manipulation Competition

In a simulation track for the IROS Robotic Grasping and Manipulation Competition, held in Daejeon, Korea October 10-12, 2016, participating teams will compete in a simulated environment. This lets teams demonstrate advanced planning and control capabilities without the need to bring hardware to the competition. A simulation framework implemented in Python using Klamp't is provided to teams. Teams will need to perform a set of tasks:

- **Bin-picking with regular objects**: lift as many balls as possible from a box and deposit them into a second box. Score 1 point for each successfully transferred object, receive a penalty of 0.5 points for each ball that is dropped outside either box
- **Picking irregular objects from a cluttered shelf**: extract as many objects as possible from a cluttered shelf and drop them inside a box. Objects are picked arbitrarily from the the YCB and Amazon datasets. For each successfully extracted object that is placed into the box 5 points are awarded.

Unlimited runs are allowed for any given task, with up to 30 minutes to complete both task. The best score for each task, maximized over all runs conducted on that task, is recorded. The presented emulator is used simulate both the Pisa/IIT SoftHand and the Reflex Hand out-of-the-box, and teams may also use it to implement custom hands.

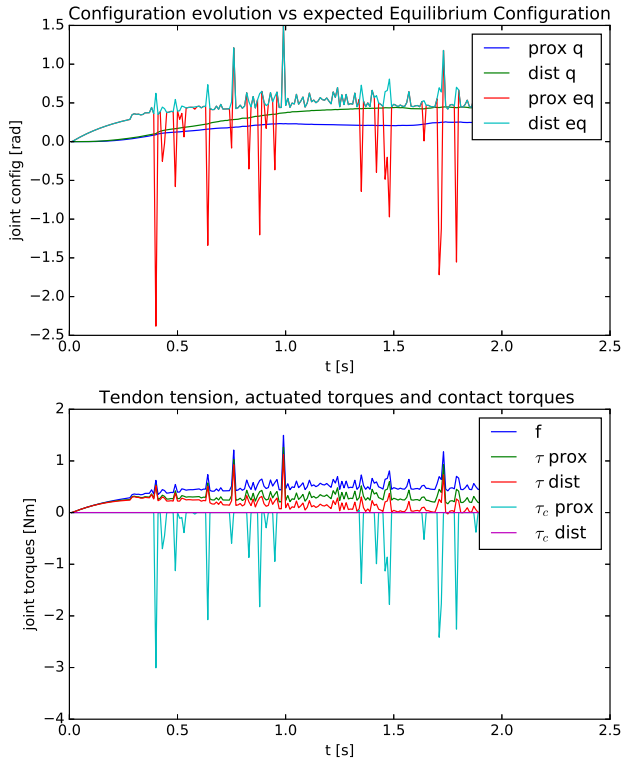

Fig. 6. A grasp experiment with the SoftHand, from the *YCB* set of the IROS Manipulation Challenge, using object `play_go_rainbow_stakin_cups_9_red` (as depicted on Fig. 1). Values for the proximal and distal joint of the index finger are shown. In the top subplot, sensed configuration is plotted against predicted configuration using eq. (5) In the bottom subplot, tendon tension is plotted against joint torques as actuated by the emulator on the underactuated joints, and against the joint torques resulting from contact forces on the index finger.

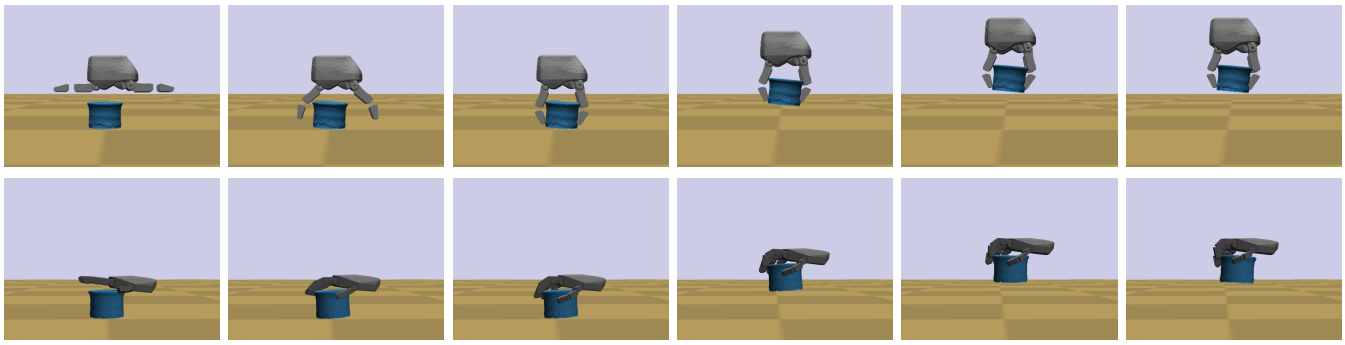

Fig. 7. A sequence showing CUHE being used to simulate the Reflex Hand (top) and SoftHand (bottom) grasping and lifting object `play_go_rainbow_stakin_cups_9_red` from the *YCB* dataset by using `simple_controller.py`, provided as example base controller in the simulation framework package for the Robot Grasping and Manipulation Competition

VI. Conclusions

A generic emulator for a large class of underactuated compliant hands has been presented. An overview of the architecture of the emulator-simulator system has been given, with details on the parameters needed to integrate a new hand model in the system. In particular, details on simulating two popular grippers thanks to the emulator and integration with the Klamp't simulator have been provided. Lastly, information on the IROS Manipulation Competition simulation track have been presented, with focus on the implementation of new underactuated compliant hands using the presented emulator. Future work may achieve better results by disabling emulation of roll-articular joints and implementing them using constraints in the underlying Linear Complementarity Problem (LCP) solver of the underlying simulator software. The emulator and its integration in the IROS Manipulation Challenge framework can be downloaded from `https://github.com/arocchi/IROS2016ManipulationChallenge`.

VII. Acknowledgements

Special thanks to Carlos J. Rosales for the discussions on the Pisa/IIT SoftHand Gazebo simulator plugin and Edoardo Farnioli for its invaluable suggestions.

References

[1] L. U. Odhner, L. P. Jentoft, M. R. Claffee, N. Corson, Y. Tenzer, R. R. Ma, M. Buehler, R. Kohout, R. D. Howe, and A. M. Dollar, "A compliant, underactuated hand for robust manipulation," *The International Journal of Robotics Research*, 2014.

[2] RightHand Robotics, "Reflex sf spec sheet," http://www.righthandrobotics.com/main:reflex, accessed: 2015-08-26.

[3] M. G. Catalano, G. Grioli, E. Farnioli, A. Serio, C. Piazza, and A. Bicchi, "Adaptive synergies for the design and control of the pisa/iit softhand," *The International Journal of Robotics Research*, vol. 33, no. 5, pp. 768–782, 2014.

[4] Robotiq, "3-finger adaptive robot gripper spec sheet," http://robotiq.com/products/industrial-robot-hand/, accessed: 2015-08-26.

[5] R. Deimel and O. Brock, "A compliant hand based on a novel pneumatic actuator," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 2047–2053.

[6] ——, "A novel type of compliant, underactuated robotic hand for dexterous grasping," *Robotics: Science and Systems, Berkeley, CA*, pp. 1687–1692, 2014.

[7] R. Ma, L. Odhner, and A. Dollar, "A modular, open-source 3d printed underactuated hand," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, May 2013, pp. 2737–2743.

[8] A. Ajoudani, S. B. Godfrey, M. Bianchi, M. G. Catalano, G. Grioli, N. Tsagarakis, and A. Bicchi, "Exploring teleimpedance and tactile feedback for intuitive control of the Pisa/IIT SoftHand," *IEEE Trans. Haptics*, vol. 7, no. 2, pp. 203–215, Apr. 2014.

[9] D. Kappler, J. Bohg, and S. Schaal, "Leveraging big data for grasp planning," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4304–4311.

[10] J. Kim, K. Iwamoto, J. Kuffner, Y. Ota, and N. Pollard, "Physically based grasp quality evaluation under pose uncertainty," *Robotics, IEEE Transactions on*, vol. 29, no. 6, pp. 1424–1439, Dec 2013.

[11] C. Eppner and O. Brock, "Planning grasp strategies that exploit environmental constraints," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, May 2015, pp. 4947–4952.

[12] A. Rocchi, B. Ames, Z. Li, and K. Hauser, "Stable simulation of underactuated compliant hands," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, 2016.

[13] G. Grioli, M. Catalano, E. Silvestro, S. Tono, and A. Bicchi, "Adaptive synergies: an approach to the design of under-actuated robotic hands," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*. IEEE, 2012, pp. 1251–1256.

[14] K. Hauser, "Robust contact generation for robot simulation with unstructured meshes," in *International Symposium on Robotics Research, Singapore*, 2013.

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

[16] C. J. Rosales, "Pisa/IIT soft hand," https://github.com/CentroEPiaggio/pisa-iit-soft-hand, accessed: 2015-08-26.

[17] A. Miller and P. Allen, "Graspit! a versatile simulator for robotic grasping," *Robotics Automation Magazine, IEEE*, vol. 11, no. 4, pp. 110–122, Dec 2004.

[18] B. León, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, *et al.*, "Opengrasp: a toolkit for robot grasping simulation," in *Simulation, Modeling, and Programming for Autonomous Robots*. Springer, 2010, pp. 109–120.

[19] R. Diankov and J. Kuffner, "OpenRAVE: A planning architecture for autonomous robotics," *Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34*, vol. 79, 2008.

[20] M. Malvezzi, G. Gioioso, G. Salvietti, and D. Prattichizzo, "Syngrasp: A matlab toolbox for underactuated and compliant hands," *Robotics Automation Magazine, IEEE*, vol. 22, no. 4, pp. 52–68, Dec 2015.

[21] M. Bonilla, E. Farnioli, C. Piazza, M. Catalano, G. Grioli, M. Garabini, M. Gabiccini, and A. Bicchi, "Grasping with soft hands," in *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 581–587.

[22] M. S. Corp, "Adams," http://www.mscsoftware.com/product/adams, accessed: 2015-08-26.

[23] M. Bonilla, D. Resasco, M. Gabiccini, and A. Bicchi, "Grasp planning with soft hands using bounding box object decomposition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, 2015, pp. 518–523.

[24] L. Birglen, "From flapping wings to underactuated fingers and beyond: a broad look to self-adaptive mechanisms," *Mechanical Sciences*, vol. 2, no. 1, pp. 5–10, 2011.