Force Reference Extraction via Human Interaction for a Robotic Polishing Task: Force-Induced Motion

1st Sara Hamdan
Dept. of Computer Science
Ozyegin University
34794 Istanbul, Turkey
sara.hamdan@ozu.edu.tr

2nd Erhan Oztop
Dept. of Computer Science
Ozyegin University
34794 Istanbul, Turkey
erhan.oztop@ozyegin.edu.tr

3rd Barkan Ugurlu
Dept. of Mechanical Engineering
Ozyegin University
34794 Istanbul, Turkey
barkan.ugurlu@ozyegin.edu.tr

***Abstract*—In this paper, a method to control a manipulator using force-induced trajectory is proposed. The trajectory is learned from an operator doing the polishing task using a tool attached to the robot's end effector. The learning process is performed by a deep neural network which is designed and trained to generate a force profile according to the states (joints' positions and velocities). The admittance control technique is utilized to make the manipulator compliant to the operator movements in the teaching mode. Spring-Damper system along with Inertia-Damper system have been studied to impose the relationship between the operator's applied force and the reaction of the manipulator. The universal robot (UR5) aside with a force sensor (OptoForce) are used to run the experiment. Robot Operation System (ROS) is used to accomplish the task in real time. The polishing task is learned and achieved by the robot itself, and the force trajectories are better followed using the Inertia-Damper system as the admittance controlling scheme.**

***Index Terms*—Admittance control, manipulator, human-robot interaction, force trajectory**

I. Introduction

For the last couple of decades, robotic manipulators have been extensively employed in factories. These robots have accomplished repetitive tasks in an agile and efficient manner, provided that the tasks are meticulously pre-engineered in a well-controlled environment. Therefore, they must remain stationary within the workflow for longer periods, causing a certain amount of decrease in the transformability of production lines. This fact appears to be a big issue when considering the fact that industrial manufacturing goes through a paradigm shift from mass production to mass customization [1].

In response to this matter, Industry 4.0 (the 4^{th} Industrial Revolution) enforces the use of robots that can adaptively collaborate with humans in a safe and dependable manner [2]–[4]. Therefore, it would be possible to blend in the robots within the human working environment as a key player to form hybrid human-robot cooperative teams [5]. In this perspective, one of the main prerequisites in achieving this goal is to ensure the stability while enhancing physical interaction capabilities.

The tasks that involve physical interaction leads to a certain energy exchange with the environment, posing technical challenges [6]–[8]. Yet, these tasks can be achieved by means of simultaneous motion and force control [9]. As stated in [10], there are two approaches to address control issues for the robots that physically interact with the environment: i) the hybrid position-force control that divides the task space into position-controlled and force-controlled subspaces [11], ii) the compliance/impedance control that manages the position-force trade-off through the active regulation of mechanical impedance [12]–[14]. The adaptability factor in impedance control schemes surely contributed to their popularity, and such controllers have been implemented to various robotic systems [15]–[17].

The energy exchange with the environment is directly linked to the management of stability vs. performance trade-off, imposing the need for certain adaptation of the rendered mechanical impedance to ensure passivity [18], [19]. To this end, several methods have been proposed; for instance, Aydin *et al.* utilized a fractional order admittance controller to ensure adaptability [20]. In another example, Gribovskaya *et al.* devised a method in which the user's intentions were anticipated to adapt the motion in accordance with the perceived motion [21]. Rozo *et al.* used machine learning to achieve transfer impedance-based behaviors to a torque-controlled robot [22]. Following a similar strategy, Dimeas and Aspragathos employed a learning algorithm for the minimization of jerk via varying damping so as to achieve effective human-robot cooperation [23].

Evidently, varying the mechanical impedance characteristics plays a pivotal role in improving the performance metrics concerning the physical interaction applications [17]–[23]. In another view, mechanical impedance adjusts the interplay between the force and motion constraints [9]. Since the motion constraints are given a priori in most robotic applications, varying the mechanical impedance could satisfy force constraints due to physical interactions.

In an alternative view, the above-mentioned objective can be fulfilled if we can synthesize feasible force constraints for a given task. To contribute toward this direction, we propose a human-to-robot skill transfer method in which the force trajectories for the robotic polishing task is obtained from a human demonstrator and then learned using a deep neural network scheme. With the addition of an admittance controller, the humans can physically manipulate the polisher that is attached to the robot end-effector to train it. In order to formulate the force trajectories in terms of position constraints, robot states

(all joint positions and velocities) are assigned as inputs to the neural network. Once the neural network is trained and able to generate force constraints in terms of robot states, the task can be achieved by using the force-induced motion controller displayed in Fig. 1.

The reminder of the paper is organized as follows: the proposed force reference extraction algorithm, including the real-time admittance controller, is explained in Section II. Experimental results are presented and discussed in Section III. Finally, the paper is concluded in Section IV.

II. Force Reference Extraction: Polishing Task

A. Experimental Setup

An industrial manipulator with six degrees of freedom (DOF) was used as the experimental testbed. A polishing tool was attached to the end-effector that allows the operator to interact with the robot. The external force applied by the operator was measured using a 6-axis Force/Torque (F/T) sensor mounted between the end-effector and the polisher.

The main blocks of the method can be seen in Fig. 1, which are: the industrial manipulator, the force sensor, the deep neural network (DNN) block, the admittance control scheme. For the industrial manipulator, the Universal Robot (UR5) manipulator was chosen due to its highly customizable properties and its suitability to optimize low-weight collaborative processes, such as picking, placing, and testing [1]. The robot is equipped with absolute encoders in each joint such that all the robot states, i.e., joint positions (θ) and velocities (ω), are readily available.

The UR5 manipulator has seven links and six revolute joints. The size of the links are given by the manufacturer and verification on the manipulator were done before using them. The forward kinematic has been derived depending on the research [24] then extra calculation has been done to take into account the sensor axes.

A high-resolution 6-axis F/T Sensor (OptoForce) was attached to the robot's end-effector in order to detect interaction forces in all three Cartesian axes (x, y and z) as well as all three rotational axes (roll, pitch, yaw or α, β, γ). A polishing tool was added then to be held by the operator and do the polishing task.

B. ROS-based Joint Velocity Control

The ROS was built as a collection of nodes that communicate using $messages$ and $topics$. The main service in ROS is the $roscore$ which provides connection information to each node and let it form a direct peer to peer connection (without central routing service) with other nodes publishing and subscribing to the same message topics. A ROS $workspace$ should be set up (made and initialize) before writing any code, the workspace (usually named $catkin\_ws$) can be imagined simply as a set of directories which contain the related set of the code [25].

To work with the UR5 robot, the ROS kinetic with Ubuntu 16.04 or ROS Indigo with Ubuntu 14.04 are recommended.

[1] www.universal-robots.com/products/ur5-robot/

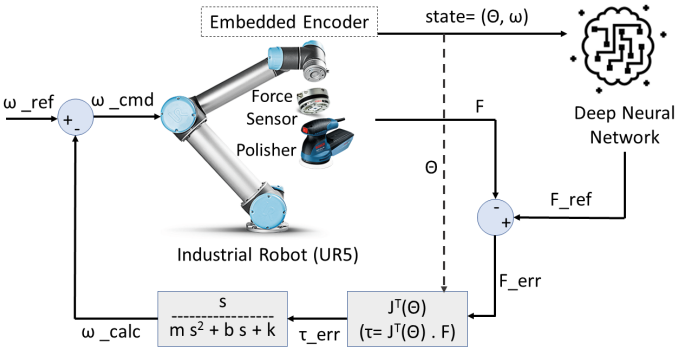

Fig. 1: Force-induced motion scheme, using a mass-spring-damper system; if $m = 0$ and $k \neq 0$ then it is a spring-damper system, if $k = 0$ and $m \neq 0$ then it is a mass-damper system. See section III for details.

The ur_modern_driver should be installed [2], then launched by $roslaunch$ to start the communication with the robot. It is also recommended in ROS to separate the code into different modules (nodes) and make them publish/subscribe to a topic. In our implementation, we created an encoder node which subscribes to the encoder topic and receives the joints position and velocity. We created another node which subscribes to the force sensor topic (which was defined and launched beforehand to communicate with the force sensor [3]) and reads the forces and moments applied on the sensor at each sampling time. Fig. 2 shows the nodes which we created to run the experiment; the ur_driver is the universal robot driver and it refers to the main code that controls the robot. The $robot\_state\_publisher$ is the encoder node, which is used by the main code to read the $joint\_states$ (the joints position and velocity). The $test\_move$ is the node that contains the trajectory that the robot should follow, the graph shows a topic named $follow\_joint\_trajectory/status$ which sends the robot current status to the node, so the node will depend on it and generate a goal trajectory and sends it by a topic named $follow\_joint\_trajectory/goal$. The node $etherdaq\_node$ is the force sensor node which runs in parallel with the ur_driver and keeps publishing the forces and moments values (with a rate of 500 Hz; The publishing speed of the F/T information, and can be changed), so the main node (which is subscribed to it) can read it any time.

C. Real-time Admittance Control Scheme

As a definition, the admittance control is a mass (inertia)-spring-damper between the target position and the actual position of the robot. It can be imagined as pushing a stick through a very viscous substance, like honey or wet sand; the more force is applied with the stick to the substance, the further the stick will move. Fig. 3 shows the mass-spring-damper system

[2] https://github.com/ros-industrial/ur_modern_driver
[3] https://github.com/OptoForce/etherdaq_ros

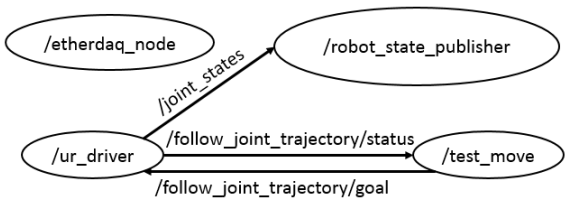

Fig. 2: ROS nodes which control the UR5 robot. The figure was built by the help of the GUI ROS plug-in rqt_graph.

and its free-body diagram. The spring force is proportional to the displacement of the mass, x (k is the spring constant), and the viscous damping force is proportional to the velocity of the mass, $v = \dot{x}$ (b is the damping coefficient). Both forces oppose the motion of the mass (m) so they are shown (in Fig. 3-(b)) in the negative x-direction. The force equation of the whole system can be written as in Eq. (1) and in Laplacian as in (2) where S is the Laplace variable. In (3) the transfer function is formulated. The spring-damper function can be found by eliminating the mass element (m), as in (4). Similarly, the mass-damper function can be found by eliminating the spring element (k), as in (5) [26].

$$F = m\ddot{x} + b\dot{x} + kx \tag{1}$$

$$F(S) = mS^2X(S) + bSX(S) + kX(S) \tag{2}$$

$$\frac{X(S)}{F(S)} = \frac{S}{mS^2 + bS + k} \tag{3}$$

$$\frac{X(S)}{F(S)} = \frac{S}{bS + k} \tag{4}$$

$$\frac{X(S)}{F(S)} = \frac{1}{mS + b} \tag{5}$$

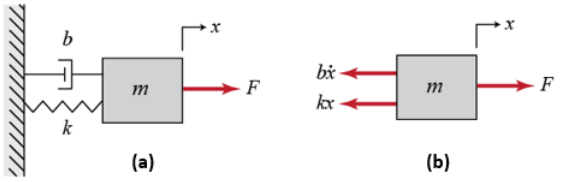

Fig. 3: (a)-Mass-Spring-Damper System, (b)-free-body diagram

D. Neural Network

Neural Network can be defined as a connectionist computational system. The normal computational systems are procedural; a program executes the code line by line. A true neural network does not follow a linear fashion; it processes the information in parallel throughout a network of nodes (neurons) [27].

We built a deep neural network (DNN), to predict the required force to be applied by the polisher at each state as a function of robot states (joint positions and velocities). The network was designed as two hidden layers as shown in Fig. 4. The number of neurons in each layer (L1 and L2) with their activation functions will be specified in accordance with the loss value while training the neural network. The input dataset, i.e. robot states, was formed by twelve features (N), and the related data was obtained by means of joint encoders. Six of them are the position of each joint (six joints), and the other six are the angular velocity of the joints. While the operator is moving the robot (generating a trajectory), the encoder will give us many data points (M) for each feature, so the input size is $N \times M$. The output of the neural network is the force and moment values which the network should predict after training, so the output size (O) is six values (forces and moments in the Cartesian space).

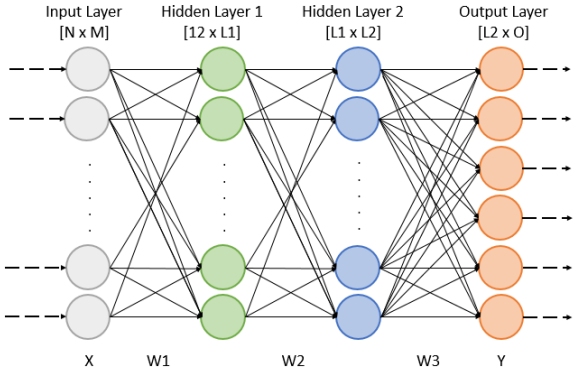

Fig. 4: Deep Neural Network Design, NxM: the shape of the input dataset, O: number of outputs, L1 and L2: number of neurons in the first and second hidden layers respectively. W1, W2 and W3: the weights between the layers.

In order to collect the DNN dataset, we made the robot operate in the admittance control mode and let the operator hold the polisher (attached to the robot) and do the polishing task following many different trajectories. During that, we collected the joint space position and velocity with a sampling time of 8 ms which is the highest sampling frequency allowed in real-time UR5 operations. In addition, we collected the force sensor readings at each time step.

After collecting a suitable amount of data and before training the neural network on that data we used a cubic spline interpolation technique to generate a function which tries to fit that data. The spline formed by a sequence of polynomial curves which are connected together (end to end) to construct a complex curve, with the condition that the formed curve is continuous and smooth [28]. The spline technique gives us the power of controlling the shape of complex curves by choosing the number of data points (called control points) and that helps in training the NN. Fig. 5 shows the result of spline which we applied on the collected velocity from one of the robot's joints (the base joint). After choosing a suitable number of control

points, we obtained a curve which sufficiently represents the data and can be used to train the neural network.

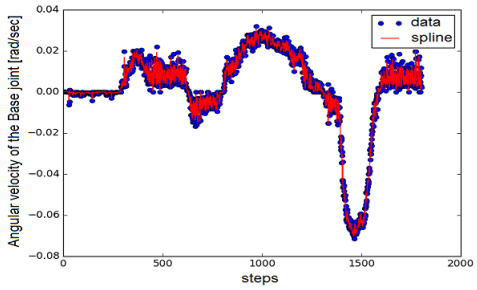

Fig. 5: Interpolation using spline curves, applied on the angular velocity of the robot's base joint.

A new dataset is formed after applying the spline technique on the collected data (position, velocity, force and moments), and that dataset is used to train the neural network. To train the NN efficiently, we split the dataset randomly into training, validation and testing datasets. The splitting has been done with the help of a Python library (SkiKit), the $test\_size$ has been choosing as 0.25, so 25% of the dataset will be used for testing and the rest for training. In accordance with the loss calculation, we edited the network design (number of neurons, activation functions and number of epochs). The loss values (MSE: mean squared errors) were recorded in each epoch for the training and validation datasets and plotted after the training finished, as shown in Fig 6. From the plot, we can see that the model has a good performance on both train and validation datasets. That performance was reached by choosing forty neurons in the first hidden layer and twenty neurons in the second one. The neurons' activation function was chosen to be rectified linear unit (ReLU), which is the most commonly used activation function in neural networks.

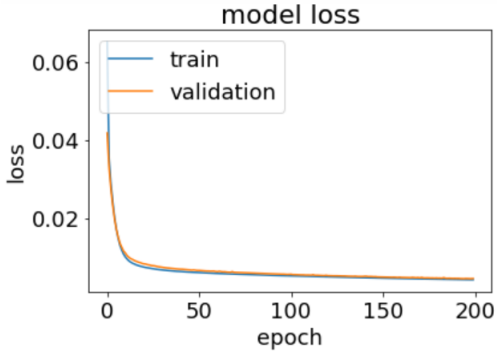

Fig. 6: The model loss on training and validation datasets.

After building the model (specifying the number of layers, number of neurons, and the activation functions), a compiling step was performed to configure the model for training. The MSE function was chosen as the objective function, and the stochastic gradient descent (SGD) was chosen as the optimizer.

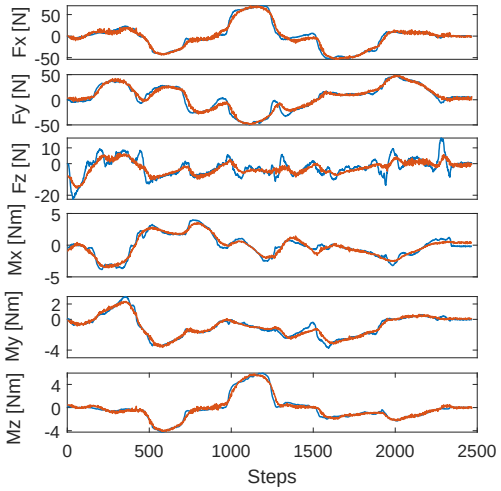

Fig. 7: Measured force/moment (in blue) and predicted force/moment (in red).

The metrics to be evaluated by the model during training and testing were set as MSE and MAE (Mean Absolute Error). We performed the training step for a suitable number of epochs (200 epochs were chosen). After completing the training step, we tested the neural network using the test part of the dataset, and that provided the predicted force and moment. A comparison between the predicted forces and the measured ones is shown in Fig. 7. The result validates that the network has favorable representation capabilities concerning the interaction forces.

III. Experiment Results

The evaluation of the proposed force-induced motion scheme was conducted experimentally using the UR5 manipulator in cooperation with a human for a robotic polishing task. The full experiment was conducted by following these steps; i) a human operator was asked to do the polishing task by holding the polisher which is attached to the robot's end effector, ii) the admittance controller accepts the applied force exerted by the operator and make the robot move compliantly in accordance with the operator's will, iii) during the motion, the robot states and interaction forces were recorded to form the input and output datasets and F_{ref} in Fig. 1 was set to zero, iv) the input dataset was interpolated by using the spline technique, and then given as the input for the designed and trained deep neural network, v) once the training was completed (within one or two minutes, depending on the trajectory), the DNN generated the force profile, i.e., F_{ref} in accordance with the robot states, vi) the force error (F_{err}) is calculated and the transposed Jacobian is used to find the torque error (τ_{err}), vii) finally we convert those generated values to angular velocity (ω_{calc}) using the admittance control scheme in Fig. 1 then it (named ω_{cmd}) commanded to the robot using the ROS script code. In this study, ω_{ref} is always set to 0 to generate force-induced motion.

The experiment was conducted by considering two approaches; in the first one, the spring-damper system was used as the admittance control part of the study (Fig. 1). In doing so, the Eq. (4) was used to create the admittance control block. To control the stiffness and the damping effect of the manipulator's motion, the spring constant (k) along with the damper coefficient (b) for each joint should be controlled. After executing many trials and evaluating the apparent effect, those values can be found as mentioned in Table I. The second approach made use of the inertia-damper system to build the admittance control block as in Eq. (5). The inertia (m) and damping (b) coefficients for each joint are displayed as in Table I. The table values - which are empirically tuned - follow the logic of giving bigger values to the bigger joints (Base, Shoulder and Elbow joints), and smaller values to the smaller joints (Wrist 1,2 and 3).

In order to generate the force-induced motion, the angular velocity reference (w_ref in Fig. 1) was set to zero; the robot starts doing the motion which was induced from the force values. While doing the motion, the resulted velocity was measured in order to compare it with the instructed ones and evaluate the performance. Fig. 8 depicts the comparison between the two velocities while using the spring damper system (force-induced velocities are shown in blue and measured ones are in green) for all of the six joints. We can see from the figure that the two curves are closely matched, despite some noisy values in all joints and more obvious in the small joints (W4,W5 and W6) which can be fixed by parameters tuning, but that was not a considerable effect on the particular polishing task (the first three joints are dominating). Moving on to the inertia-damper system, Fig. 9 displays that the two velocity profiles nearly matched as well. Moreover, the noise was much reduced in all joints which led to increased compliance in this mode when compared to the spring-damper mode. There is still a noticeable noise in the small joints but can be ignored as mentioned before.

Spring (k) - Damper (b) System	**Mass (m) - Damper(b) System**
$k_1 = 30$, $b_1 = 500$	$m_1 = 100$, $b_1 = 300$
$k_2 = 30$, $b_2 = 500$	$m_2 = 100$, $b_2 = 300$
$k_3 = 30$, $b_3 = 500$	$m_3 = 100$, $b_3 = 300$
$k_4 = 20$, $b_4 = 200$	$m_4 = 80$, $b_4 = 100$
$k_5 = 20$, $b_5 = 200$	$m_5 = 80$, $b_5 = 100$
$k_6 = 20$, $b_6 = 200$	$m_6 = 80$, $b_6 = 100$

TABLE I: system coefficient values which are empirically tuned

IV. Conclusion

In this study, we presented a method to extract the force reference profile from an operator movement -while doing the polishing task- and generate a force-induced motion for the manipulator. The extraction process was achieved by means of a deep neural network which we designed and trained based on the robot states (joint positions and velocities). The trained network predicted the force profile which was then subsequently followed by the manipulator to accomplish the task autonomously. The admittance control technique was used, which introduced compliance between the applied force and the motion of the robot. Two different approaches were used to design the admittance control scheme; the spring-damper approach and the inertia-damper approach. Comparing the performance of them showed that the second one is more suitable to use to accomplish the polishing task by the manipulator since it led to less fluctuation.

The proposed method succeeded in controlling the robot using an extracted force reference to achieve the robotic polishing task. In our future work, the system coefficients will be tuned by a reinforcement learning algorithm to find the optimum values to accomplish the task.

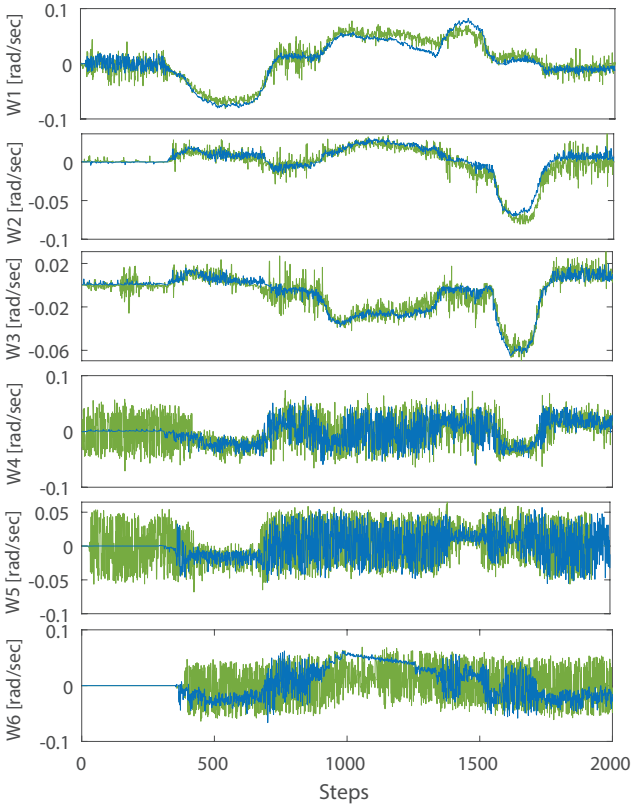

Fig. 8: force-induced commanded velocities (in blue) and measured ones (in green) using spring-damper system

Acknowledgements

This research is supported by the EU and conducted as a part of the H2020 project CoMRAde, with the grant number 753219. The authors thank Arcelik Atolye 4.0 and Ercan Gunduz for providing the hardware setup.

References

[1] F. Griffiths, and M. Ooi, "The fourth industrial revolution - Industry 4.0 and IoT [Trends in Future I&M]," in *IEEE Instrumentation and Measurement Magazine*, vol. 21, no. 6, 2018, pp. 29-43.

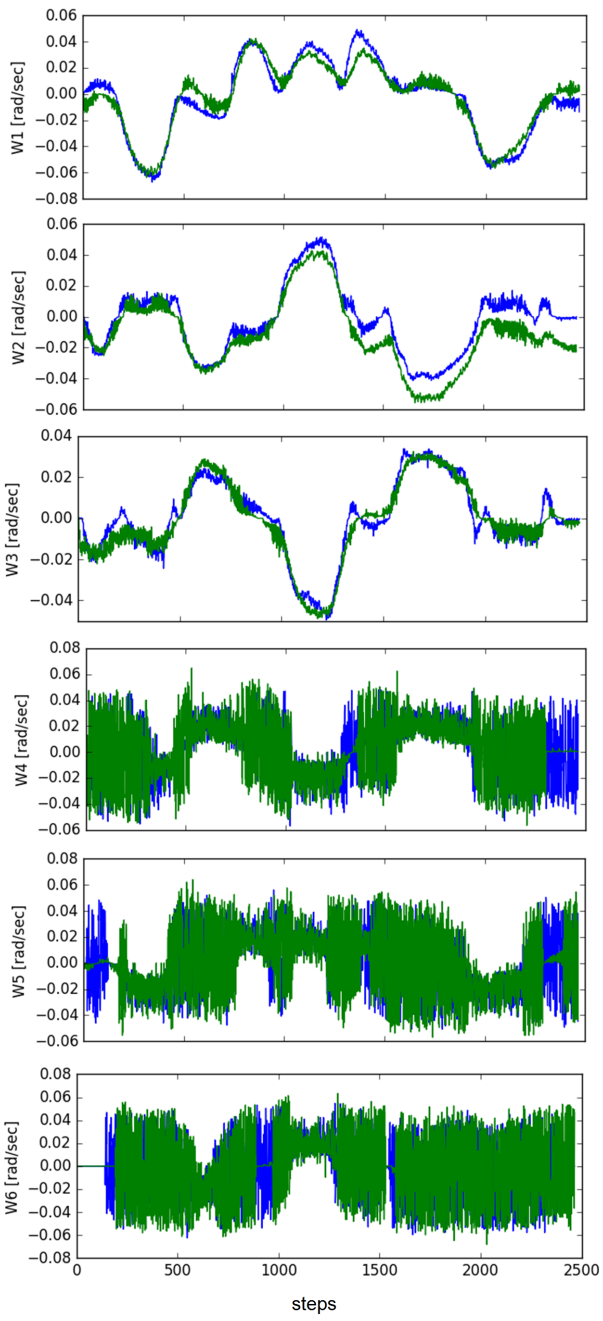

Fig. 9: force-induced commanded velocities (in blue) and measured ones (in green) using inertia-damper system

[2] A. Sciutti, M. Mara, V. Tagliasco, and G. Sandini, "Humanizing Human-Robot Interaction: On the Importance of Mutual Understanding," in *IEEE Technology and Society Magazine*, vol. 37, no. 1, 2018, pp. 22-29.

[3] M. Zinn, O. Khatib, B. Roth, and J. K. Salisbury, "Playing it safe [human-friendly robots]," in *IEEE Robotics and Automation Magazine*, vol. 11, no. 1, 2004, pp. 12-21.

[4] A. De Santis, B. Siciliano, A. De Luca, and A. Bicchi, "An atlas of physical humanrobot interaction ," in *Mechanism and Machine Theory*, vol. 43, no. 3, 2008, pp. 253-270.

[5] M. A. Shehadeh, S. Schroeder, A. Richert, and S. Jeschke, "Hybrid teams of industry 4.0: A work place considering robots as key players," in Proc. of the *IEEE Int. Conf. on Systems, Man, and Cybernetics*, Banff, Canada, 2017, pp. 95-100.

[6] S. D. Eppinger, and W. P. Seering, "Three dynamic problems in robot force control," in *IEEE Robotics and Automation Magazine*, vol. 8, no. 6, 1992, pp. 751-758.

[7] T. Valency, and M. Zacksenhouse, "Accuracy/robustness dilemma in impedance control," in *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 125, no. 3, 2003, pp. 310-319.

[8] G. Zeng, and A. Hemami, "An overview of robot force control," in *Robotica*, vol. 15, no. 1, 1997, pp. 473-482.

[9] O. Khatib, "A unified approach for motion and force control of robot manipulators: The operational space formulation," in *IEEE Transactions on Robotics and Automation Magazine*, vol. 3, no. 1, 1987, pp. 43-53.

[10] C. Ott, R. Mukherjee, and Y. Nakamura, "Unified impedance and admittance control," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, Alaska, US, 2010, pp. 554-561.

[11] M. Raibert and J. Craig, "Hybrid position/force control of manipulators," in *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 103, no. 2, 1981, pp. 126-133.

[12] M. T. Mason, "Compliance and force control for computer controlled manipulators," in *IEEE Transactions on System, Man and Cybernetics*, vol. 11, no. 6, 1981, pp. 418-432.

[13] N. Hogan, "Impedance control: an approach to manipulation: part I-theory," in *ASME Journal of Dynamic Systems, Measurement and Control*, vol. 107, no. 1, 1985, pp. 1-7.

[14] J. K. Salisbury, "Active stiffness control of a manipulator in cartesian coordinates," in Proc. of the *IEEE Int. Conf. on Decision and Control*, Albuquerque, US, 1980, pp. 95-100.

[15] W. S. Kim, B. Hannaford, and A. K. Bejczy, "Force-reflection and shared compliant control in operating telemanipulators with time delay," in *IEEE Transactions on Robotics and Automation Magazine*, vol. 8, no. 2, 1992, pp. 186-185.

[16] S.-H. Hyon, "Compliant terrain adaptation for biped humanoids without measuring ground surface and contact forces," in *IEEE Transactions on Robotics*, vol. 25, no. 1, 2009, pp. 677-688.

[17] B. Ugurlu, C. Doppmann, M. Hamaya, P. Forni, T. Teramae, T. Noda, and J. Morimoto, "Variable ankle stiffness improves balance control: experiments on a bipedal exoskeleton," in *IEEE/ASME Transactions on Mechatronics*, vol. 21, no. 1, 2016, pp. 79-87.

[18] K. Kronander, and A. Billard, "Stability considerations for variable impedance control," in *IEEE Transactions on Robotics*, vol. 32, no. 5, 2016, pp. 1298-1305.

[19] M. J. Kim, W. Lee, J. Choi, G. Chung, K.-L. Han, I.-S. Choi, C. Ott, and W. K. Chung, "A Passivity-based nonlinear admittance control with application to powered upper-limb control under unknown environmental interactions," in *IEEE/ASME Transactions on Mechatronics*, Accepted, 2019.

[20] Y. Aydin, O. Tokatli, V. Patoglu, and C. Basdogan, "Stable physical human-robot interaction using fractional order admittance controls," in *IEEE Transactions on Haptics*, vol. 11, no. 3, 2018, pp. 464-475.

[21] E. Gribovskaya, A. Kheddar, and A. Billard, "Motion learning and adaptive impedance for robot control during physical interaction with humans," in Proc. of the *IEEE Int. Conf. on Robotics and Automation*, Shangai, China, 2011.

[22] L. Rozo, S. Calinon, D. Caldwell, P. Jimenez, and C. Torras, "Learning Collaborative Impedance-Based Robot Behaviors," in Proc. of the 27^{th} *AAAI Conf. on Artificial Intelligence*, Washington, US, 2013, pp. 1422-1428.

[23] F. Dimeas, and N. Aspragathos, "Reinforcement learning of variable admittance control for human-robot co-manipulation," in Proc. of the *IEEE Int. Conf. on Intelligent Robots and Systems*, Hamburg, Germany, 2015.

[24] K. Kufieta,"Force Estimation in Robotic Manipulators: Modeling, Simulation and Experiments," Ph.D. dissertation, Dept. of Engineering Cybernetics NTNU Norwegian University of Science and Technology, 2014.

[25] B. Gerkey, M. Quigley, and W. Smart, Programming Robots with ROS: A Practical Introduction to the Robot Operating System, O'Reilly Media, 1st edition, 2015.

[26] M. Gopal, Control Systems, Tata McGraw-Hill Education, 2008.

[27] D. Shiffman, The Nature of Code, chapter 10 Neural Networks, 2012.

[28] S. McKinley and M. Levine, Cubic Spline Interpolation, College of the Redwoods, 1998.