2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)
April 4-8, 2022, Edinburgh, UK

A Desktop-sized Platform for Real-time Control Applications of Pneumatic Soft Robots

Brandon J. Caasenbrood[1], Femke E. van Beek[1], Hoang Khanh Chu[1], and Irene A. Kuling[1]

***Abstract*— In recent years, there have been many modeling and control advances in the field of soft robotics, which resulted in a growing interest in practical, real-time applications. To further enable these developments, we present a desktop-sized testing and development platform intended for fast, precise, and reliable (closed-loop) control of pneumatic soft robots. The Soft Robotics Control-unit (SRC) is fully compatible with Matlab, Simulink, and Unity, allowing for real-time control tasks of various complexity. The system's performance is tested in three use-cases: model-based controller design, soft haptic feedback in virtual-reality, and tele-operation of a soft gripper. To promote the use of the SRC in the soft robotics community, all presented material is fully open-sourced such that it is easily reproducible by students or researchers from any technical background.**

I. Introduction

In the past decade(s), the diversity in soft robotic design has been growing rapidly, ranging from the popular PneuNets [1] and artificial soft muscles [2] to a vast collection of bio-inspired systems [3]–[6]. Although soft robotic design and actuation is rather diverse in nature, a majority of soft robotic systems rely on fluidics or pneumatics. Pneumatics have the advantage of being readily available, yet, there exists a stark contrast between the ease of soft robotic design and the burden of developing a tailored pneumatic control systems. In many cases, researchers are forced to develop a custom control interface [3], [7]–[9], as commercially-available solutions are limited. In particular, with the increasing development in model-based controllers for soft robotics, there exists an increasing demand in versatile platforms that allow for complex control strategies that are nowadays standardized in rigid robotics. Some examples include: motion control, impact and robust control, locomotion, tele-operation, haptic feedback, and machine and reinforcement learning. We believe that such control strategies for soft robotics could advance faster if researchers would be able to use a standardized development platform.

In addition to enabling faster development, the standardization of control systems would allow for a better performance description of new soft robotic designs. Currently, descriptions of new soft robots often present numbers on for instance response times [10]. However, these performance values partially depend on the soft robotic design, the hardware, and the software that is used to control the pneumatic networks (e.g, PD [11], [12] and bang-bang control [13]–[15]). As such, standardization of control hardware and software could make full system characterization more informative. To aid in these aims, we present a fully integrated solution for real-time control of pneumatic soft robotics systems.

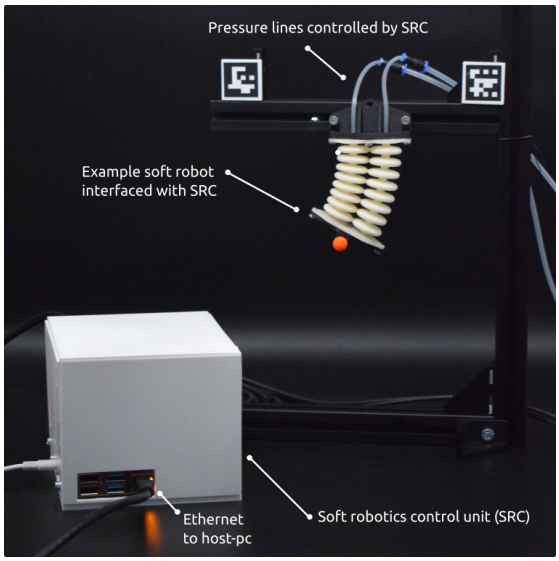

Fig. 1. Overview of the Soft Robotics Control-unit (SRC) connected to a planar bending soft robot with two pressure inputs. Through a TCP/IP server, the pneumatic control unit can communicate with a host-pc running MATLAB/Simulink in real-time – thus allowing for fast (closed-loop) control and simultaneous data acquisition.

A. State-of-the-art

The development of pneumatic control architectures is perhaps not new, and the aim of this work is not to compete with existing system, but rather focus the branch of fast, precise (closed-loop) control applications. A popular option for pneumatic control, especially for newer researchers to the field of soft robotics, is the 'Soft Robotics Toolkit' [11], [12]. This platform has been a major way to democratize soft robotics, including a vast catalogue of soft robotic designs, models, and low-level controls. Other democratization initiatives are the FlowIO development platform [14], [15] and the Programmable-Air platform [16]. Although these initiatives offer great prototyping capabilities, they are limited in more advanced control settings where small onset latencies, high flow-rates, and computational flexibility are essential for real-time control applications. Booth et al. have produced a very small, fast, and wearable-robot control system [13]. Although their onset latencies do come close to our requirements, and their solution is very elegant for on-robot-wearable form factors, their flow rates (0.1 $\mathrm{L/s}$)

[1]B.J. Caasenbrood, F.E Beek, H. Khanh Chu, I.A. Kuling are with the Faculty of Mechanical Engineering, Dynamics and Control Group, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands. Email: b.j.caasenbrood@tue.nl, f.e.v.beek@tue.nl, h.chu.khanh.hoang@student.tue.nl, and i.kuling@tue.nl

2022 IEEE 5th International Conference on Soft Robotics (RoboSoft) | 978-1-6654-0828-8/22/$31.00 ©2022 IEEE | DOI: 10.1109/ROBOSOFT54090.2022.9762137

are perhaps not sufficient for fast actuation of larger-sized robots. Another platform, which is perhaps closely related to the work presented here, is the work of Young et al. [8]. Yet we focus instead on a system that primarily supports single-board computers rather than micro-controllers, e.g., Arduino Due, which might limit higher bandwidths, advanced on-board computations, and data acquisition of multiple sensor nodes. Another open-source system that allows real-time control is the PneumaticBox [9], yet their system does not support proportional pressure regulation.

Rather than focusing on small-scale and autonomous soft robots in which compactness is key, we focus on a class of soft robotic systems that are suitable in a classic industrial setting, e.g., manipulation, pick-and-place, and general motion control. We also envision use-cases for haptics and tele-operation, which must satisfy tight delay constraints and high-bandwidths. The haptic sense (i.e., sense of touch) reliably detects latencies exceeding 50 ms between motor actions and haptic feedback [17], and tele-operation systems suffer from instabilities in the presence of delays [18]. Similar instability issues might occur in model-based control strategies for soft robotics, where small delays can diminish the intrinsic passivity of the systems. To our knowledge, a system meeting these requirements does not exist yet, hence we offer an open-source alternative to the existing state-of-the-art systems.

B. Contribution

In this work, we present an open-source Soft Robotics Control Unit (SRC) for pressure-driven soft robots, with a particular focus on fast, precise, and accurate closed-loop applications. Compared to previous hardware, our main contributions include:

- Desktop-sized, modular control platform for real-time control of pneumatic soft robots;
- Low-level controller suited for piezo-actuated pressure valves – RMS errors ≤ 1 kPa and delay ≤ 20 ms.
- High-frequency control loop with bandwidths up to 500 Hz (up to 12 individual pneumatic channels);
- Sensor support via standardized I^{2}C Qwiic connection (up to individual 6 sensor channels)[1];
- Several options for fast and easy software interfacing via MATLAB, Simulink, and Unity.

The paper is organized as follows. First, we give an overview of the control platform, detailing the hardware, software, and its compatibility for interfacing with other software. Next, we discuss a low-level controller to improve the system's performance. Lastly, we show the performance of the SRC in three soft robotic use cases with different levels of control complexity.

II. Platform Overview

Here we briefly detail the hardware and software components of the SRC. To support the soft robotics community, all material presented in this work (i.e., hardware, software, documentation, and use cases) is made publicly available at the following repository[2]. We would like to stress that, although some components are custom-made, our open-source resources make it easy for researchers or students from any technical background to reproduce the SRC.

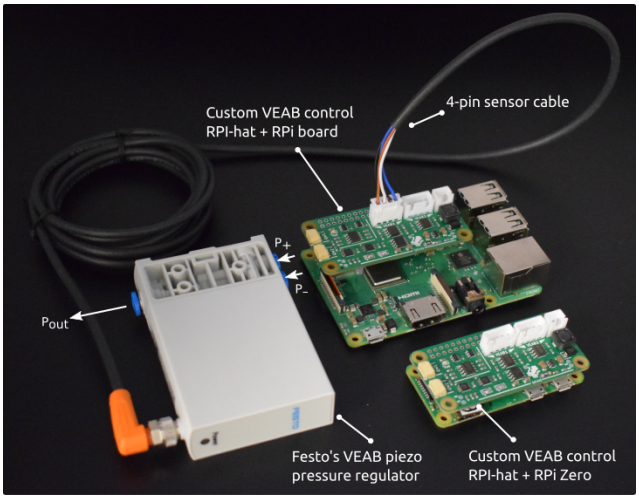

Fig. 2. Images of the SRC's custom real-time controller hat stacked on a Raspberry Pi 3 (above) and on a Raspberry Pi Zero (below). The RPi hat is compatible with the Raspberry Pi 3, 4, and Zero. It has two Qwiic I^{2}C ports for additional sensors, two ports for the Festo's VEAB pressure regulators, and an onboard 24V buck-boost converter. For the RPi 4, up to six VEAB controller hats can be stacked in a single system.

A. Hardware overview

In the following section, we will detail the inner hardware of the SRC control unit. The system consists of a single-board computer (Raspberry Pi 4, 2GB RAM), a custom Raspberry Pi Hat that interfaces between the computer and two pressure regulators. There exist various options for pressure regulation, but we opted for Festo's three-way proportional valve (VEAB-L-26-D7-Q4-V1-1R1) with an active range of $-0.1 < P(t) \leq 0.1$ MPa differential pressure and flow rate of $0.075 \leq Q(t) \leq 0.33$ $\mathrm{L/s}$. The custom RPi hat can support upto two VEAB regulators each, and is versatile to fit many board computer or micro-controller options. The enclosure is fully 3D-printed, and can be fabricated using a standard FDM printer.

The custom circuit board has a small footprint of 30×60 mm and can be stacked as a Raspberry Pi Hat. To interface with the VEAB regulators, the circuit board has two Digital-to-Analog converters (MicroChip, MCP4725) and two Analog-to-Digital converters (Texas Instruments, ADS1014) that send and receive data from the regulators, respectively. Given the specifications of the DACs and ADCs, the system has, under ideal conditions, a control and measurement resolution of $e = \pm 0.048$ kPa. Both the ADCs and DACs are paired with non-inverting opamps (Texas Instruments, LM358S) such that the voltage-levels (in this case 3.3V) are compatible with various board computers,

[1]The Raspberry Pi 4 supports up to 6 unique I^{2}C busses. Each I^{2}C bus can support multiple sensor; however, each sensor ICs must have a unique I^{2}C address on the same bus to avoid cross-talk.

[2]`github.com/chukhanhhoang/SoftRoboticSetupFesto`

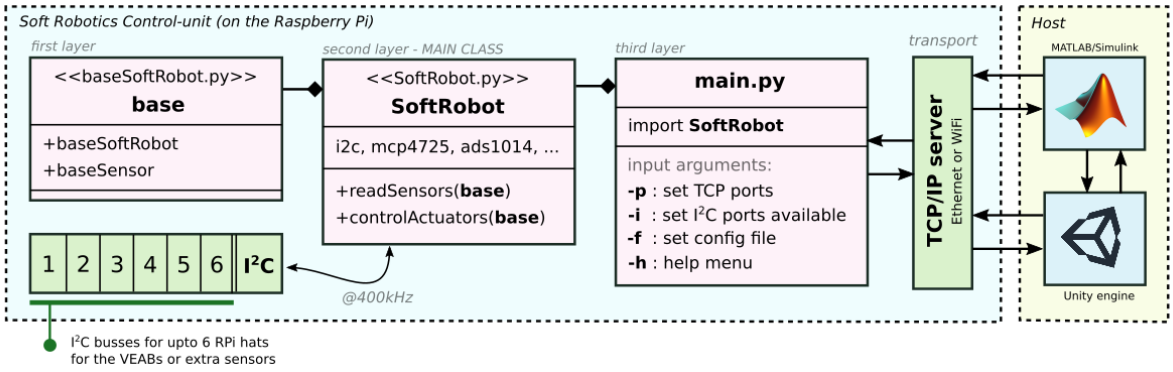

Fig. 3. Schematic overview of the software architecture running on the Raspberry Pi 4, and its interfacing options. The software architecture consists of three consecutive layers that build upon each other. All sensor and hardware I^2C communication occurs in the second layer; whereas the last layer acts as an optional layer for changing configuration parameters. Lastly, the python code starts a TCP/IP server to interface with Matlab/Simulink and Unity externally.

e.g., any Raspberry Pi system or Nvidia Jetson Nano. The on-board array of ADCs and DACs can be accessed via I^2C. To enable a stackable design, we exploit the available six I^2C busses on the Raspberry Pi 4 – thus allowing for connecting twelve regulators simultaneously. Furthermore, each board is equipped with two I^2C Qwiic connectors that allow for additional sensor communication at 400 kHz. Our open-sourced PCB design files can be used to order the boards from most PCB manufacturing facilities that allow for in-house PCB fabrication and component placement.

B. Software architecture

The main software of the setup, written in Python, is designed in three layers to optimize for speed, convenience, and versatility. The setup can read multiple sensors and control several VEAB regulators simultaneously, as well as communicate with other devices via TCP/IP. An illustrative diagram of the software architecture is given in Figure 3.

The first layer of the software provides two base classes: *baseSoftRobot.py* and *baseSensor.py*. The class *baseSoftRobot.py* sets up the multi-processing environment that handles the TCP/IP communication for the array of VEAB regulators and sensors. Multi-processing is used here to process data in parallel, improving not only the speed of the system but also the timing precision. The class *baseSensor.py* serves as a wrapper for other sensor classes. Both classes act as parents for other classes in the next layer(s).

The second layer facilitates all communication between the Raspberry Pi and the SRC real-time controller hat. First, necessary libraries are imported. Next, sensor classes are written as children of the *baseSensor.py* class, and in each sensor class, a function called *ReadSensor* must be implemented when using sensors. Finally, the class *SoftRobot*, inheriting all functions of *baseSoftRobot.py*, is assembled from all software parts relating to the TCP/IP communication, regulators, and sensors. Thus, the class *SoftRobot* acts as the main class of the software architecture that encompasses the user's requirement.

The third layer is dispensable and added for the convenience of users. This layer is a Python script that takes the arguments and initializes a *SoftRobot* object accordingly. Changing the parameters of the software (i.e., number of sensors and actuators, sampling frequency, setting the TCP port) is no longer an arduous task of re-writing the code.

C. MATLAB, Simulink, and Unity interfaces

One of the platform's key features is its versatility in terms of programming interface. First, in MATLAB, we developed an Object-Oriented class `SRC.m`, which can be found on the repository. The class allows for easy communication with the hardware platform through an array of integrated functions. Alternatively, the Simulink interface provides the same functionality as the MATLAB interface, but it provides GUI-based programming. The base sets up the TCP/IP communication with the SRC, and additional functionalities can be added using auxiliary Simulink blocks.

For control scenarios related to human-robot cooperation, we explore the Unity game engine. Therefore, a Unity package for TCP/IP communication is developed based on the work of Weiland [19]. Important parameters are exposed to Unity's interface, so that they can be set by the user without the need for changes in the source code.

We provide working examples of the three interface options (i.e., MATLAB, Simulink, and Unity) to further support the use-cases presented in section III. However, let it be clear that any interface that allows for TCP/IP communication is compatible with minimal changes to the software.

D. Low-level controller design

To enhance the pneumatic performance of the SRC, we extend beyond the standard PD- and bang-bang control architecture as often presented literature [11]–[15]. As such, we propose a novel low-level controller that ensures a low tracking error and delay w.r.t. a dynamic reference. First, the dynamics of the intrinsic VEAB pressure regulation together with the pneumatic network of the soft robot can be approximated using a first-order dynamical system of the form:

$$\Sigma : \begin{cases} \dot{x} = Ax + \phi(u), \\ p = x, \end{cases} \tag{1}$$

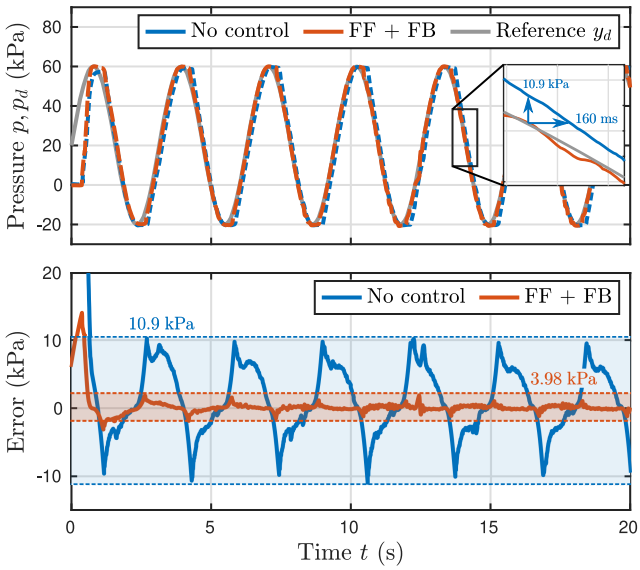

Fig. 4. Performance results of the control unit under dynamic testing in which a feed-forward (FF) and feedback (FB) controller is employed for accurate and fast pressure regulation. Here we observe a substantial performance increase using the proposed control law given in (2)-(3).

Fig. 5. A few snapshots of the PneuNet when subjected to a regulated sinusoidal pressure signal $p(t)$. Left: *(a)* Placement of the inertial sensor for real-time curvature sensing that are communicated via I^2C to the SRC.

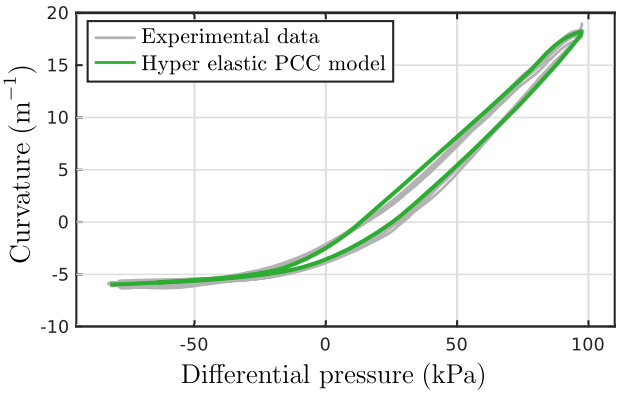

Fig. 6. System identification results of the PneuNet actuator using experimental data – and the model-fit of the hyper-elastic PCC model. The system is subjected to a periodic sine input, we observe a hysteresis loop, hyper-elasticity, and self-contact.

where x denotes the internal pressure state of the pressure regulator, u the control input, p the pressure output of the system, and $\phi : \mathbb{R} \to \mathbb{R}$ a nonlinear input map due to the piezo-actuation of the VEAB regulator. To be more specific, the (monotonic) mapping $\phi(u)$ describes the local nonlinearity of the regulators for pressure signals close to ambient pressure. This behavior (similar to a dead zone) is important to include in the model, as it is the lead cause of tracking errors when switch from positive to negative differential pressures, or when considering high-frequent fluid flows. The control objective here is to ensure that the VEAB regulator closely follows the reference trajectory, denoted here by p_d. As such, we wish to ensure that the tracking error goes to zero after sufficient time passes, i.e., $\lim_{t\to\infty} p \to p_d$. To achieve this, we propose a combination of feed-forward (FF) and feedback (FB) control:

$$u = \hat{\phi}^{-1}(\nu), \tag{2}$$

$$\nu = F_d \dot{p}_d + p_d - K_p(p - p_d), \tag{3}$$

where K_p is the proportional gain of the feedback controller, F_d the differential gain of the feed-forward controller, and $\hat{\phi}^{-1}$ the approximation of the inverse mapping to counteract the deadzone[3] such that it satisfies $\phi \circ \hat{\phi}^{-1} \approx 1$ (i.e., $\phi(\hat{\phi}^{-1}(x)) \approx x$ which attempts to counteract the piezo nonlinearity). Note that if the signal $\dot{p}_d$ is unavailable, one can employ derivative filtering scheme.

E. System performance measurements

To test the performance of the full system, we connect the SRC to the PneuNet shown in Figure 5, and considered a dynamic pressure trajectory $p_d = 20 + 40\sin(t)$. To illustrate the effectiveness of the low-level controller, we also compared the open and closed-loop performance. The open-loop controller is equivalent to setting the input of the controller to the reference signal, i.e., $u = p_d$. The performance results of the low-level controller are shown in Figure 4, where the RMS error and maximum error are ± 0.35 kPa and ± 1.91 kPa, respectively. All reported pressures are relative to atmospheric pressure. In contrast, the open-loop system has a RMS error and maximum error of ± 3.98 kPa and ± 10.93 kPa, respectively. Regarding the phase shift, we observe an estimated difference of ± 160 ms and ± 45 ms in open-loop and closed-loop, respectively. Given these results, we show that the low-level controller reduces both the tracking error and delay substantially. Please note that the system parameters of Σ are unique and depend for instance on the tube length, volume, and the type of regulator. For optimal performance, tuning of the gains might be required for every soft robot.

[3]The deadzone function $\phi : \mathbb{R} \to \mathbb{R}$ is assumed to be a monotonically increasing function, such that its inverse exists.

III. Use Cases for Soft Robotics

In this section, we highlight the performance of the SRC in terms of precision and delay, and its versatility. To do so, we explore three use cases: *i)* system identification and model-based control of a soft robot, *ii)* tele-operation using a soft gripper, *iii)* and soft haptic feedback using Virtual Reality (VR). To improve transparency, all the study-cases are provided in the supplementary material of the paper, and the software is made publicly available at the repository under the folder `./src/examples`.

A. System identification and model-based control of PneuNet

In the first study case, we primarily focus on model-based strategies for a bending soft robot seen in Figure 5. Our aim here is two-fold: *i)* to employ system identification to derive a dynamic model, and *ii)* to utilize the dynamic model for control. As such, fast and low-latency pressure regulation is important. To measure the state of the system, an Inertial-Measurement Unit (InvenSense, MPU9250) is attached to the end-effector of the soft robot (see Figure 5). The sensor is connected via I^{2}C to the SRC and the data acquisition is programmed in the Python class `SoftRobot.py` (see Section. II-B). Based on preliminary tests, we propose an ansatz for the (nonlinear) dynamics using an in-extensible planar Piece-wise Constant Curvature (PCC) model that accounts for hyper-elastic stiffness and elastic self-contact. It is worth mentioning that there exists a rich literature on PCC models, and the reader is referred to [4], [20]–[22] for more detail and to [23] for the numerical implementation. Let us introduce the state $q := \kappa$ with κ the (constant strain) curvature. Then, the dynamic model takes the form:

$$M(q)\ddot{q} + C(q,\dot{q})\dot{q} + P(q,\dot{q}) + G(q) = Hp, \qquad (4)$$

where $M(q)$ is the generalized inertia matrix, $C(q,\dot{q})$ the Coriolis matrix, $P(q,\dot{q}) = k(q)q + \mu\dot{q}$ the hyper-elastic and visco-elastic forces with $k(q)$ a nonlinear stiffness and $\mu > 0$ a damping coefficient, $G(q)$ the gravitational forces, H a pressure map, and p the control input (i.e., pressure). To model the hyper-elastic stiffness and self-contact, we propose the following nonlinear state-dependent function:

$$k(q) = \alpha_1 e^{-\min(\alpha_2 q, 0)} + \alpha_3 \ (q - q_c)^{\alpha_4} \cdot \min(q, 0), \qquad (5)$$

where $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ are hyper-elastic stiffness coefficients, and q_c the joint position when self-contact of the soft body occurs. In the identification procedure, we subject the soft robot to a periodic pressure input $p = 10 + 80\sin(2t)$ kPa. The resulting dynamics converge to a periodic solution, on which we fit our dynamic model. The system identification results of the ansatz in (4)-(5) using the periodic experimental data show a good match between the model and measurement, as can be seen in Figure 6.

Now, using the aforementioned model, we propose a model-based controller to control the soft robot's state:

$$p = H^{-1}\left[P(q,\dot{q}) + G(q) - K_p e - K_i \int e(\tau)d\tau\right], \qquad (6)$$

where K_p, K_i are proportional and integrator gains, respectively; and $e := q - q_d$ the tracking error between the state and the desired trajectory. The model-based controller is a combination of gravity-stiffness compensation together with a PI-controller. The model-based controller above is programmed using MATLAB/Simulink and communicates via TCP/IP with the control unit (at 200 Hz). Considering a described dynamic trajectory $q_d = 8 + 8\sin(2t)$, the tracking performance of the model-based control law is shown in Figure 7, where we observe good tracking performance with a maximum steady-state error of $e = \pm 1.42$ m^{-1}. These

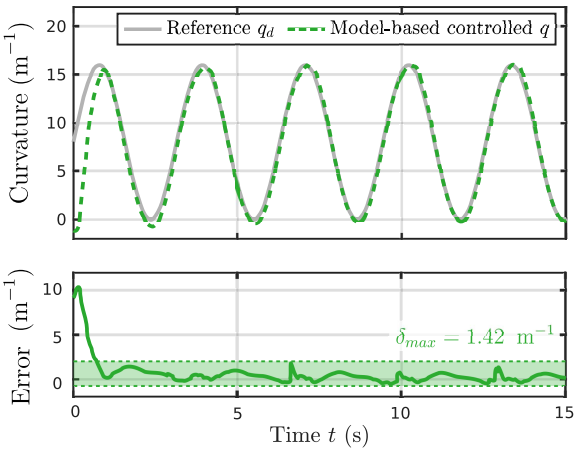

Fig. 7. Experimental results of the proposed model-based controller in (6). We observe that the model-based controller can provided good tracking performance – the steady-state error is $\delta_{\max} \leq 1.42$ (m^{-1}).

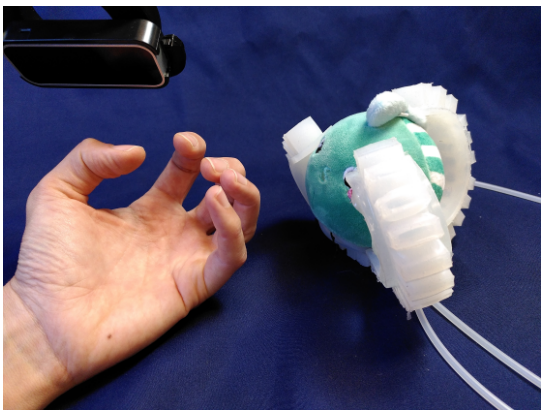

Fig. 8. Overview of the tele-operation system in use case 3. The angles of the user's thumb, index and middle finger are captured by the Leap Motion camera, and mapped to pressures of the three-finger Pneunet gripper.

results demonstrate the SRC's versatility as a platform for the development and testing of model-based controllers.

B. Tele-operation of soft gripper

In the next use case, we focus on a tele-operation scenario in which a user's hand movements are mapped to a remote three-fingered soft PneuNet gripper (DragonSkin 10) as shown in Figure 8. In this particular setting, minimizing latency outweighs tracking performance here, as minimal latency will improve human-robot transparency [24] and allow for better system stability [18]. To track the user's hand, a Leap Motion tracker (Ultraleap) is used, positioned on a stand above a desk. The joint angles of the user's thumb, index and ring finger are registered by Unity, and received by MATLAB/Simulink using the TCP/IP communication package.

To match the Leap Motion system, the refresh rate is set to 120 Hz. To measure the effect of the additional communication layers consistently, a ramp is used as a representative reference signal. The reference signal is communicated from Unity through Simulink to the SRC, and the measured signal

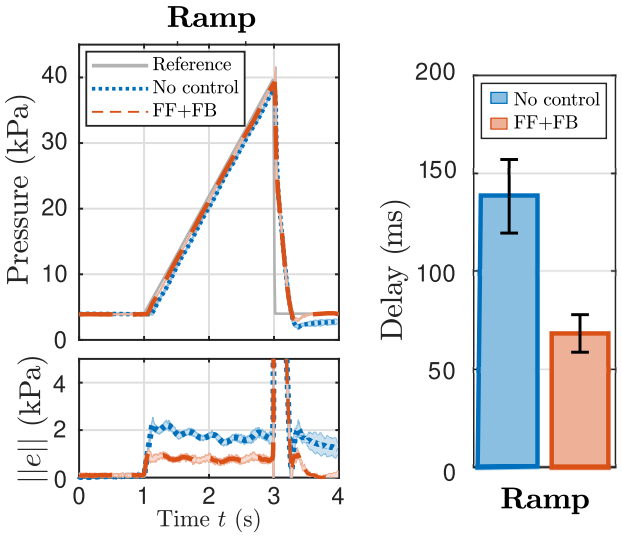

Fig. 9. Left: Performance for tele-operation in Unity. The top row shows performance without (blue) and with our low-level controller (red). The bottom row shows the difference (i.e. vertical error) between the reference (gray) and measured signals (blue/red). In both figures, the 95% confidence interval is indicated with shaded regions, but the ten repetitions show such comparable responses that it is hard to see the intervals. Right: Time delays (i.e, horizontal error), indicating the horizontal distance between the reference and measured pressures shown on the left. Time delays for the low-level controller are much smaller than open-loop control delays.

is send then back to Unity and stored. The measurement is repeated ten times, with and without the low-level controller (FF+FB, see (2)-(3)). The controller is also tuned a-priori, to better suit the use case. As illustrated in Figure 9, the open-loop latency is considerably high (138 ± 19 ms) for the current setup. However, with the low-level controller, the delay is substantially smaller (68 ± 9 ms), albeit still above the human perceptual limit. This illustrates that the proposed system, under the appropriate control conditions, can be a suitable candidate for exploring soft robotic tele-operation in lab conditions.

C. Haptic feedback in Virtual Reality

In the next use case, we employ soft robotics to provide haptic (i.e., touch) feedback to users in a virtual interface. Adding haptic feedback to Virtual-Reality (VR) scenarios has the potential to create better immersion and interaction with virtual objects [25]. However, such benefits only arise if the feedback perceived is congruent with the other signals in the scene. Therefore, the aim here is to test if the SRC can maintain fast feedback under the additional complexity of VR, and to validate actuators with smaller volumes in retrospect to previous use cases. We again use the Leap Motion tracker, however, it is now attached to a VR headset (HTC, Vive Pro VR) for hand tracking. For soft robotic haptic feedback, small cylinders with a single air chamber are designed (11x7 mm) and composed of Dragonskin 10. Silicone finger sleeves (Xutong) are used to attach the cylinders to the thumb, index and middle finger of the user's right hand, as shown in Figure 11. In the VR environment, users can interact with the virtual interface, and haptic effects are generated based on finger positions. The array of haptic effects are composed of ramps, steps, and sine-wave primitives.

To match the sampling rate of the VR headset, a refresh rate of 90 Hz is chosen. The same procedure as described in the previous use case are used to test the haptic primitives, in this use case ten times per controller setting. The results in Figure 10 show that the VR use case operates below perceptually noticeable delays (i.e., $\leq$50 ms). We do see peaks in (vertical) errors, when the reference shows large discontinuities due to tight control gains for minimal system delay rather than small vertical errors. Nevertheless, these peaks are small and short compared to the reference, thus having little effect on the user experience. Together, these results indicate that SRC is a viable option for haptic feedback in a VR setting.

IV. Conclusion

In this work, we present a desktop-sized testing and development platform purposefully designed for real-time control applications of pneumatic soft robots. Using a low-level controller that compensates for the intrinsic nonlinear dynamics of the piezo-actuated regulators, we further enhance the base performance of the closed-loop pneumatic system – in both response time ($\leq$50 ms) and tracking error ($\leq$ 1 kPa). One key contribution of this work is the platform's versatility by allowing up to twelve regulated pressure ports and an easy-to-program control interface that allows for various soft robotic systems, sensors, and control use-cases. To demonstrate its versatility and the system's performance in advanced control settings, the system is experimentally corroborated using three different study-cases: model-based control of a soft robot, soft haptics in VR, and remote tele-operation of a soft gripper. In future work, we will also develop custom sensor breakout boards and corresponding programming tools, e.g., data acquisition for strain and optical sensors, which further enrich the versatility of the platform. These additional hardware systems and any further development will be maintained on the open-repository in the future.

References

[1] B. Mosadegh, P. Polygerinos, C. Keplinger, S. Wennstedt, R. F. Shepherd, U. Gupta, J. Shim, K. Bertoldi, C. J. Walsh, and G. M. Whitesides, "Pneumatic networks for soft robotics that actuate rapidly," *Advanced Functional Materials*, vol. 24, no. 15, pp. 2163–2170, 2014.

[2] F. Schmitt, O. Piccin, L. Barbé, and B. Bayle, "Soft robots manufacturing: A review," *Frontiers in Robotics and AI*, vol. 5, 2018.

[3] A. D. Marchese, R. K. Katzschmann, and D. Rus, "A Recipe for Soft Fluidic Elastomer Robots," *Soft Robotics*, vol. 2, no. 1, pp. 7–25, 2015.

[4] R. K. Katzschmann, C. D. Santina, Y. Toshimitsu, A. Bicchi, and D. Rus, "Dynamic motion control of multi-segment soft robots using piecewise constant curvature matched with an augmented rigid body model," *RoboSoft 2019 - 2019 IEEE International Conference on Soft Robotics*, no. February, pp. 454–461, 2019.

[5] K. Nakajima, H. Hauser, T. Li, and R. Pfeifer, "Exploiting the Dynamics of Soft Materials," vol. 5, no. 3, pp. 339–347, 2018.

[6] V. Falkenhahn, T. Mahl, A. Hildebrandt, R. Neumann, and O. Sawodny, "Dynamic Modeling of Bellows-Actuated Continuum Robots Using the Euler-Lagrange Formalism," *IEEE Transactions on Robotics*, vol. 31, no. 6, pp. 1483–1496, 2015.

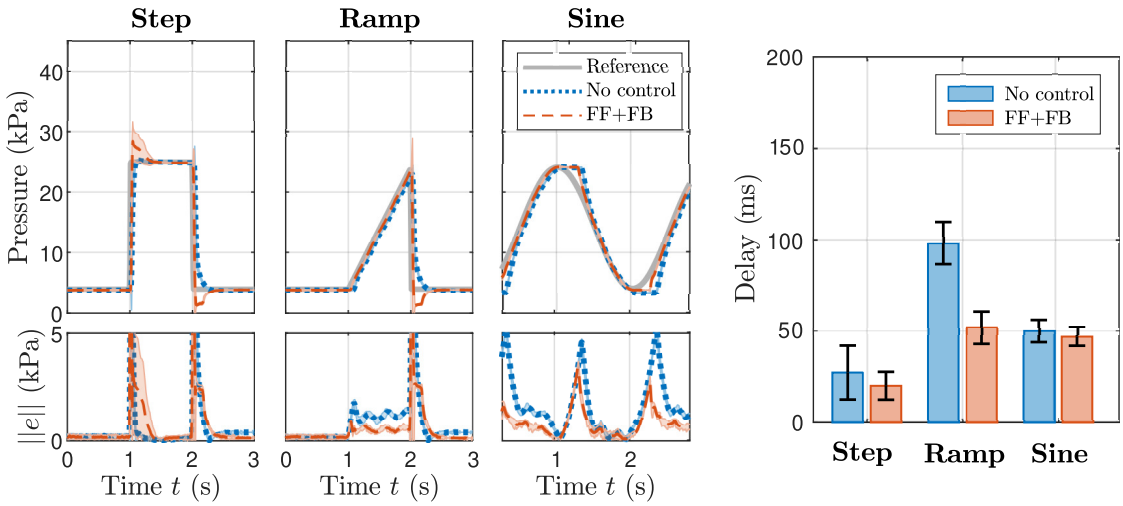

Fig. 10. Left: performance of the Haptic VR Feedback use case primitives, as recorded in Unity. For the full caption, see Figure 9. The controller is tuned to minimize delay, which results in larger overshoots compared to no control. For visibility, the tops of the peaks are cut off. Right: The time delays. For all primitives, the low-level controller reduces the delays below the perceptual detection threshold.

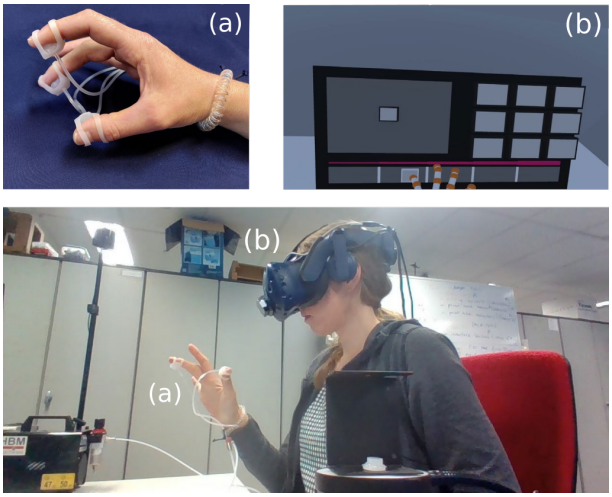

Fig. 11. Bottom: User wearing a headset while exploring an interface in use case 2: haptic feedback in VR. The interface provides visual, auditory, and haptic feedback. Top: *(a)* The soft robotic tactile feedback units, which provide (real-time) pressure feedback to the user's finger tips. *(b)* The VR interface, which provides congruent visual, auditory, and haptic feedback.

[7] P. Polygerinos, S. Lyne, Z. Wang, L. F. Nicolini, B. Mosadegh, G. M. Whitesides, and C. J. Walsh, "Towards a soft pneumatic glove for hand rehabilitation," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, 2013, p. 1512–1517.

[8] T. R. Young, M. S. Xavier, Y. K. Yong, and A. J. Fleming, "A control and drive system for pneumatic soft robots: Pneusord," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2822–2829.

[9] "Robotics: The PneumaticBox control system for soft hand control," Mar 2022. [Online]. Available: https://www.robotics.tu-berlin.de/menue/softwaretutorials/pneumaticbox

[10] R. F. Shepherd, F. Ilievski, W. Choi, S. A. Morin, A. A. Stokes, A. D. Mazzeo, X. Chen, M. Wang, and G. M. Whitesides, "Multigait soft robot," *Proceedings of the National Academy of Sciences*, vol. 108, no. 51, pp. 20 400–20 403, 2011. [Online]. Available: https://www.pnas.org/content/108/51/20400

[11] A. A. Zheng Wang, Panagiotis Polygerinos and A. Campo. Fluidic control board. Last visited: 20-09-2021. [Online]. Available: https://softroboticstoolkit.com/book/control-board

[12] D. P. Holland, C. Abah, M. Velasco Enriquez, M. Herman, G. J. Bennett, E. A. Vela, and C. J. Walsh, "The soft robotics toolkit: Strategies for overcoming obstacles to the wide dissemination of soft-robotic hardware," *IEEE Robotics and Automation Magazine, Special Issue on Open Source and Widely Disseminated Robot Hardware*, vol. 24, no. 1, pp. 57–64, 2017.

[13] J. W. Booth, J. C. Case, E. L. White, D. S. Shah, and R. Kramer-Bottiglio, "An addressable pneumatic regulator for distributed control of soft robots," *2018 IEEE International Conference on Soft Robotics, RoboSoft 2018*, no. April, pp. 25–30, 2018.

[14] A. Shtarbanov. Flowio platform. Last visited: 20-09-2021. [Online]. Available: https://www.softrobotics.io

[15] ——, "Flowio development platform – the pneumatic "raspberry pi" for soft robotics," in *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*, no. 479. New York, NY, USA: Association for Computing Machinery, 2021.

[16] Programmable-air. Last visited: 20-09-2021. [Online]. Available: https://www.programmableair.com

[17] T. Kaaresoja, S. Brewster, and V. Lantz, "Towards the temporally perfect virtual button: Touch-feedback simultaneity and perceived quality in mobile touchscreen press interactions," *ACM Trans. Appl. Percept.*, vol. 11, no. 2, 2014. [Online]. Available: https://doi.org/10.1145/2611387

[18] T. Sheridan, "Space teleoperation through time delay: review and prognosis," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5, pp. 592–606, 1993.

[19] T. Weiland. Tcp-udp networking. Last visited: 29-09-2021. [Online]. Available: https://github.com/tom-weiland/tcp-udp-networking

[20] B. J. Caasenbrood, A. Y. Pogromsky, and H. Nijmeijer, "Dynamic modeling of hyper-elastic soft robots using spatial curves," in *IFAC-PapersOnLine*, vol. 53, 2020, pp. 9238–9243.

[21] A. D. Marchese and D. Rus, "Design, kinematics, and control of a soft spatial fluidic elastomer manipulator," *The International Journal of Robotics Research*, vol. 35, no. 7, pp. 840–869, 2016.

[22] C. Della Santina, R. K. Katzschmann, A. Bicchi, and D. Rus, "Model-based dynamic feedback control of a planar soft robot: trajectory tracking and interaction with the environment," *International Journal of Robotics Research*, vol. 39, no. 4, pp. 490–513, 2020.

[23] B. J. Caasenbrood, "Sorotoki - an open-source soft robotics toolkit for matlab," github.com/BJCaasenbrood/SorotokiCode, 2020.

[24] A. Toet, I. A. Kuling, B. N. Krom, and J. B. F. van Erp, "Toward enhanced teleoperation through embodiment," *Frontiers in Robotics and AI*, vol. 7, 2020.

[25] B. Hannaford and A. M. Okamura, *Haptics*. Springer Berlin Heidelberg, 2008, pp. 719–739.