Supplementary material: Design and Control of Compliant Tensegrity Robots through Simulation and Hardware Validation

KEN CALUWAERTS[1,2], JÉRÉMIE DESPRAZ[1,3], ATIL IŞÇEN[1,4], ANDREW P. SABELHAUS[1,5], JONATHAN BRUCE[1,6], BENJAMIN SCHRAUWEN[2] AND VYTAS SUNSPIRAL[1,7]

[1]NASA Ames Research Center, USA
[2]Reservoir Lab, Electronics and Information Systems department, Ghent University, Belgium
ken.caluwaerts@ugent.be, benjamin.schrauwen@ugent.be
[3]Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Switzerland
jeremie.despraz@epfl.ch
[4]Oregon State University, USA
iscena@onid.oregonstate.edu
[5]Berkeley Institute of Design, University of California Berkeley, USA
apsabelhaus@berkeley.edu
[6]USRA/University of California Santa Cruz, USA
jbruce@soe.ucsc.edu
[7]SGT Inc., USA
vytas.sunspiral@nasa.gov

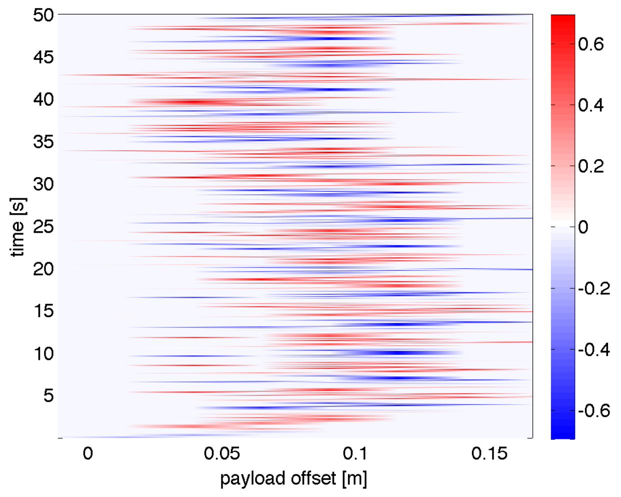

Figure 1: Intensity of the feedback signal of an inner spring as a function of the payload position relative to the center of mass and time during a single 50s run.

I. Reactive Controller Feedback Signal

Figure 1 shows the intensity of the feedback signal of an inner spring-cable assembly (i.e. a connection between the payload and the outer shell). The tensegrity robot (generic tensegrity icosahedron with payload) performs a series of rolls during a 50s simulation using the reactive algorithm. This figure highlights the signal periodicity and the constant offset between the tensegrity's center of mass and the payload position, as desired to create enough torque to flip the robot. The apparent diffuse signal stems from the shaking of the payload during locomotion. If we were to plot the average offset over multiple runs as a function of time, this would appear as a straight vertical line. The color shows the intensity of the feedback signal and the periodicity of the signal (the robot rolls at around 0.32Hz in this example).

II. Prototype capabilities

The underactuated ReCTeR robot is capable of folding, deploying and rolling. Fig. 2 shows the robot rolling in a stepwise manner through large shape deformations and a robustness test. The sequence of actuator positions (which correspond to spring-cable rest lengths) were hand-tuned for the rolling to demonstrate the capabilities of the robot.

The bottom sequence shows the robot being dropped from a 0.3m height (lowest end cap) onto an actuated end cap. We have dropped the ReCTeR robot many times from heights up to 0.5m without mechanical or electrical failures.

III. Learning a Matsuoka oscillator with Physical Reservoir Computing

This section presents the main equations for the learning algorithm presented in Section IV.3 of the main text. The observations (sensor values) of the physical structure (which we consider analog to neural activity) are given by:

$$\boldsymbol{m}(t) \quad = \quad \mathrm{vec}\begin{pmatrix} 1 \\ \boldsymbol{f}'(t) \\ \boldsymbol{f}'(t-\Delta) \\ \cdots \\ \boldsymbol{f}'(t-k\Delta) \end{pmatrix}, \tag{1}$$

where $k = 9$ is the number of time delay steps used and vec makes a vector from a matrix by concatenating the columns. As we showed in our previous work, the delays allows the feedback to filter out sensor noise, and peaks due to impacts. The $\boldsymbol{f}'(t)$

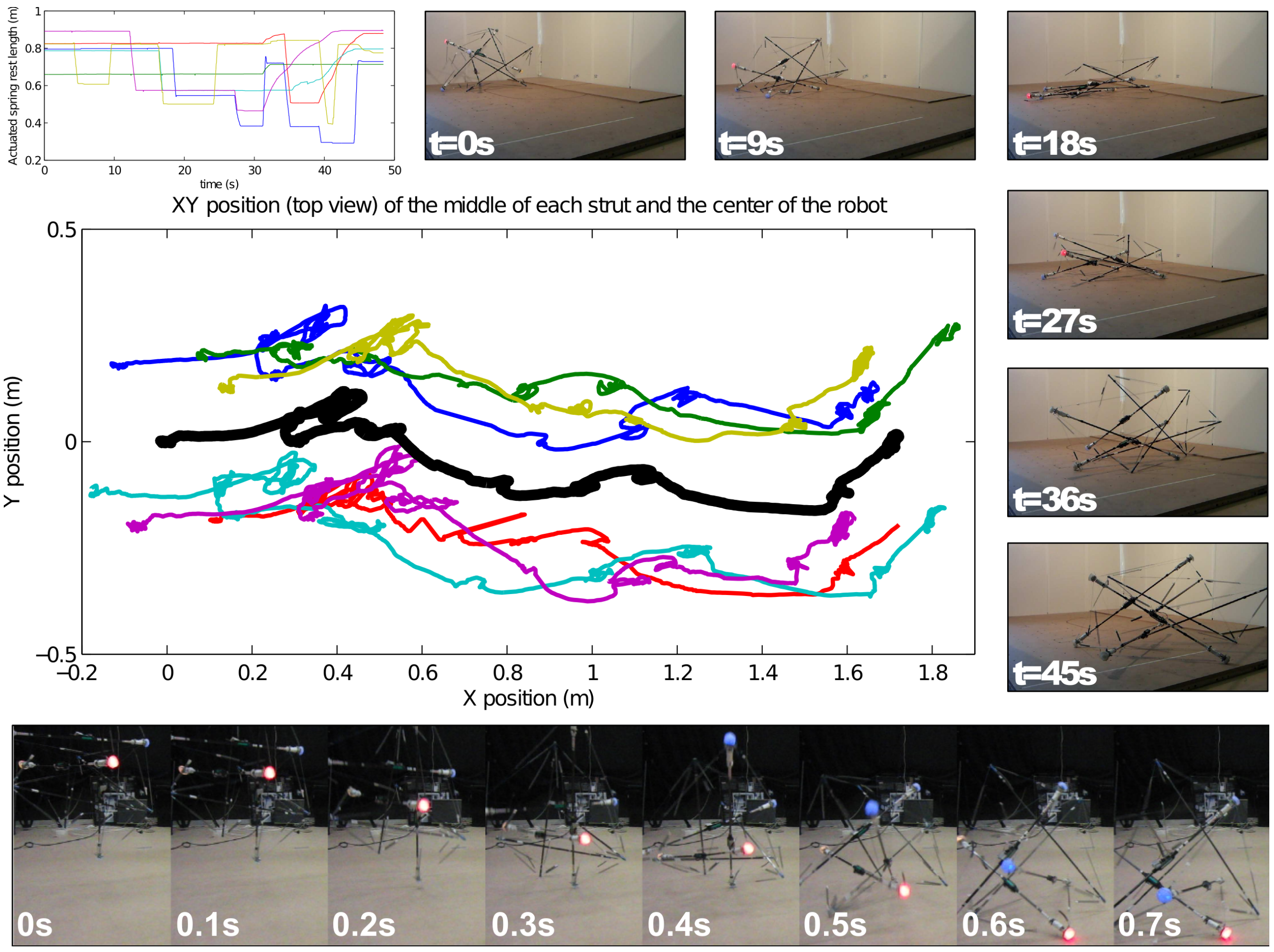

Figure 2: Top: ReCTeR stepwise rolling through large shape deformations. The sequence of actuator positions (top left plot) was found manually to show the robot's capabilities. The images show the robot at various times during the sequence. The large plot is a top view of the trajectories of each strut's center (thin lines) and the robot's center (thick line). The robot performs a full roll along one side. During the roll, we forced the robot to fold and deploy (t=18s). The actuator position plot shows that the actuation phases are short (below one second), indicating that the robot can change the rest lengths of the actuated springs fast (over 0.3m/s). After 31s and 35s, two actuators are turned off, to show the passive deploying of the structure (by backdriving the spindle motors). Bottom: 0.3m drop test showing the compliance and robustness of the robot. At 0.3s an actuated end cap hits the ground. All sensors and actuators were enabled during the experiment.

vectors are the sensor values of the strain gauge transducers. Due to the mechanical and electrical design, the sensor values will be a nonlinear function of the spring-cable assembly tensions and spring configuration:

$$f'(t) \quad = \quad o(f(t), \boldsymbol{P}), \tag{2}$$

where $\boldsymbol{P} = [\boldsymbol{p_0}\, \boldsymbol{p_1} \ldots]$ are the coordinates of end caps and o is a non-linear function of the tension and the configuration of the robot. It can be seen that o can be written as

$$o(f(t), \boldsymbol{P}) \quad = \quad \zeta f(t) \cos(\beta), \tag{3}$$

where ζ is a scaling factor and β is the angle between the spring-cable assembly and the sensor. Differently from the simulation results, we did not estimate the derivative of the sensor values as this can be cumbersome and is not required by the technique. To remain true to the simulation results, we did not use the other available sensors on the robot (e.g. IMU).

To increase the actuator efficiency, the motor signals (rest lengths of the spring-cable assemblies) $\boldsymbol{\ell} = [\ell_0\, \ell_1 \ldots]^T$ are a low passed linear combination of the observations:

$$\boldsymbol{\ell}(t) \quad = \quad 0.9\boldsymbol{\ell}(t-\Delta) + 0.1\boldsymbol{r}(t), \tag{4}$$

where $\boldsymbol{r}(t)$ represents the mix of open loop and feedback signals. This is implemented with a standard PID position controller in hardware.

The network update and output equations are given by:

$$\rho \quad = \quad \frac{1}{1+\tau t} \text{if } \frac{1}{1+\tau t} > 0.2 \text{ else } 0 \tag{5}$$

$$\boldsymbol{r}(t) \quad = \quad \rho \boldsymbol{r}^{target}(t) + (1-\rho)\boldsymbol{W}(t)\boldsymbol{m}(t), \tag{6}$$

with a learning rate parameter $\tau = 0.05$. Initially $\boldsymbol{r}(t) = \boldsymbol{r}^{target}(t)$, which means that the motor signals are fully open loop. After learning $\boldsymbol{r}(t) = \boldsymbol{W}(t)\boldsymbol{m}(t)$, which means that the motor signals are a linear combination of the sensor values (closed loop).

At each time step during learning ($\rho > 0$) the weights $\boldsymbol{W}$ are updated using the Recursive Least Squares equations while $\rho > 0$:

$$\boldsymbol{L}(t) \quad = \quad \frac{\boldsymbol{\Gamma}(t)\boldsymbol{m}(t)}{1 + \boldsymbol{m}(t)\boldsymbol{\Gamma}(t)\boldsymbol{m}(t)} \tag{7}$$

$$\boldsymbol{\Gamma}(t+\Delta) \quad = \quad \boldsymbol{\Gamma}(t) - \frac{\boldsymbol{\Gamma}(t)\boldsymbol{m}(t)\boldsymbol{m}^T(t)\boldsymbol{\Gamma}(t)}{1 + \boldsymbol{m}^T(t)\boldsymbol{\Gamma}(t)\boldsymbol{m}(t)} \tag{8}$$

$$\boldsymbol{e}(t) \quad = \quad \boldsymbol{y}^{target}(t) - \boldsymbol{W}(t-\Delta t)\boldsymbol{m}(t) \tag{9}$$

$$\boldsymbol{W}(t) \quad = \quad \boldsymbol{W}(t-\Delta t) + \boldsymbol{L}(t)\boldsymbol{e}(t), \tag{10}$$

where $\boldsymbol{\Gamma}(0) = \boldsymbol{I}$. The vector $\boldsymbol{r}^{target}(t)$ contains the oscillator state at time t.

References