Cable driven sensing simulation

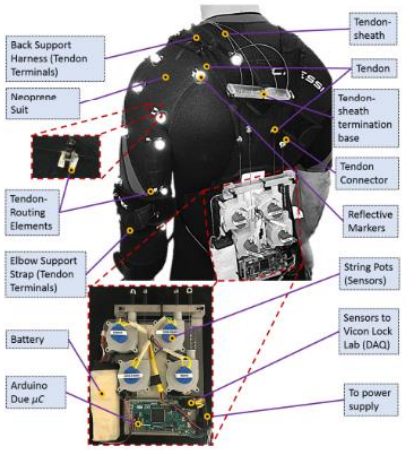

Fig. 2. Overview of the sensing framework prototype for the shoulder exosuit with a description of the electro-mechanical and mechanical elements.

Varghese, R.J., Lo, B.P.L. and Yang, G.Z., 2020. Design and prototyping of a bio-inspired kinematic sensing suit for the shoulder joint: Precursor to a multi-dof shoulder exosuit. IEEE Robotics and Automation Letters, 5(2), pp.540-547.

Background:

For soft exoskeletons to properly assist their user's movement, they need to accurately estimate the orientation of the arm. This is more challenging than with rigid exoskeletons, since they allow for a greater range of motion (which is a good thing of course!). Encoders of a soft exo's actuator relate to the movement in a heavily non-linear way, which don't map to joint angles in a straightforward way.

Simulating the tendon dynamics can capture these non-linearities and help kickstart any method that aims to decode it.

This demo is inspired by the work of Rejin Varghese, who developed such a sensing suit.

Description:

This example simulation is composed of two parts, using a model of a human shoulder, to which several sensing tendons are attached. First the joint angle space is systematically swept, and the tendon lengths are recorded and associated with the current joint configuration (shoulder_sensing_fit.py). Then a SVM regressor is fitted to this dataset, after which it can be used for inferring what the current joint angles are given the cable lengths (shoulder_sensing_predict). The prediction is displayed in the form of a transparent element pointing in the direction as estimated by the SVM model.

When using the prediction script, the transparent sphere is connected to the fingers of the model, and moving it will move the whole hand around. The prediction of the shoulder angle is visualized by a blue transparent reproduction of the upper arm, which with a well fitted model will mostly overlap with the original red arm. To move the target, double left click it, then use Ctrl+Right click drag to move it around in the vertical direction, or Ctrl+Shift+Right click drag to move it along the horizontal plane.

Goals:

- Look into the code implementation, and interpret the plots generated by shoulder_sensing_fit.py.

- Currently 4 cables are used to make the prediction. How would reducing, or increasing the amount of cables influence the quality of prediction? You can try ignoring the signals from some of the cables when you fit and use the SVM regressor to reduce the number of inputs, or edit the file titled "05_nonlinear_sensing_visu.xml" to add a 5th cable.
- What about the elbow? How many cables do you think you'd need to estimate that joint? How is it different compared to the shoulder?
- Familiarise yourself with the MuJoCo documentation to answer questions you might have about available elements for the MJCF. The documentation is available at this link:

https://mujoco.readthedocs.io/en/stable/XMLreference.html

Bonus goals:

- What if you have some noise in your sensor readings? You can use the random noise generation from libraries such `numpy.random` to simulate this. Do you have ideas how you could reduce its affect? What side effects could your approach have?