The body as a reservoir: locomotion and sensing with linear feedback

Ken Caluwaerts, Benjamin Schrauwen
Reservoir Lab - Electronics and Information Systems Department - Ghent University
B-9000 Ghent, Belgium
Email: ken.caluwaerts@ugent.be, benjamin.schrauwen@ugent.be

***Abstract*—It is known that mass-spring nets have computational power and can be trained to reproduce oscillating patterns. In this work, we extend this idea to locomotion and sensing. We simulate systems made out of bars and springs and show that stable gaits can be maintained by these structures with only linear feedback. We then conduct a classification experiment in which the system has to distinguish terrains while maintaining an oscillatory pattern. These experiments indicate that the control of compliant robots can be simplified if one exploits the computational power of the body's dynamics.**

***Keywords*—morphological computation, embodiment, reservoir computing, tensegrity, central pattern generator**

I. Introduction

It is known that neural circuits can gain universal computational power through appropriate static feedback and output [4]. If the system itself is rich enough, it is often sufficient to use only a linear readout layer, as the system can provide the necessary non-linearities itself.

Hauser et al. [1] applied this idea to random mass-spring networks in two dimensions, instead of typical neural networks. They showed that relatively small networks (approx. 10 point masses) can reproduce stable limit cycles, such as the Van der Pol oscillator, when linear feedback is used.

Basically one sees the body as a dynamical system that provides computational power and one then tries to exploit these features by using appropriate input and output transformations, while leaving the system itself unchanged. The Reservoir Computing (RC) approach [7] is very similar: one starts with a random recurrent neural network and instead of modifying the network itself, only the readout layer is trained.

In this paper we try to bring the ideas from [1] and [4] closer to robotics, by verifying that locomotion can be induced in tensegrity-like structures through linear feedback. We then show that sensing is also possible by making the system discern two types of terrain.

A. Tensegrity structures

The simplest extension of the spring-mass approach to robotics is to replace the point masses by spheres. Such structures do not resist compressive forces due to gravity very well and are hard to build.

Instead, we replace the point masses by stiff bars and thus create tensegrity-like structures. These structures tend to have a good mix of and strength and flexibility. We use the term tensegrity-like, as real tensegrity structures can be built out of only bars and (possibly elastic) strings, while in our case springs that resist both pushing and pulling are used. Creating large random tensegrity structures is not trivial [6] and is left as future work, but we expect similar computational power as these tensegrity-like structures due to the non-linear dynamics.

The springs used in this work are non-linear. The force acting along a spring is given by [1]:

$$F = k_1 d + k_3 d^3 + p_1 v + p_3 v^3 + F_{fb} \tag{1}$$

in which d corresponds to length of the spring minus its resting length and v to the velocity. The force F_{fb} generated by the linear feedback is a linear combination of the spring lengths. The parameters k_1, k_3, p_1 and p_3 are chosen randomly to create a rich pool of dynamics and hence to increase computational power, while assuring that the robot is stiff enough to avoid collapsing under gravity.

II. Methods

The simulated robots were created by randomly positioning bars within an ellipsoid with a minimum distance of 50 cm between the bars. Next, a Delaunay triangulation is performed on the endpoints of the bars (the spring attachments). A spring is added for each lattice of the triangulation.

The state of the system is defined as the lengths of the springs (minus the initial length). The output of the system is a linear combination of the system's state and only these weights are trained. A subset of the springs are actuated (force applied along the spring) and the magnitudes of the applied force (the feedback, F_{fb}) are random, fixed linear combinations of the system's output with additive Gaussian white noise (GWN).

All simulations were done using the Open Dynamics Engine with a time step of 1 ms and controller frequency of 100 Hz.

III. Locomotion

Locomotion is an active research topic in the compliant robot domain. Hauser et al. [1] showed that spring-mass networks can maintain a limit cycle with only linear feedback. Paul et al. [5] developed gaits for tensegrity structures and found relatively simple actuation patterns. We combine these results to produce a robot that generates the actuation patterns for its proper locomotion.

In a first set of experiments, we train the robot by modifying the weights from the system's state to the output as in [1] to generate a desired pattern at the output.

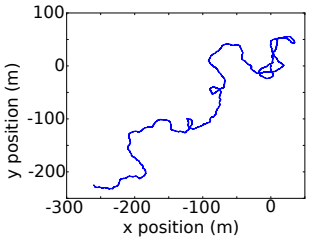

Fig. 1. Path traveled by the robot during 10100 seconds (100 s training) after learning a simple quadratic oscillator. The position of the robot is taken to be the mean of the positions of all the bars. The robot started at the origin.

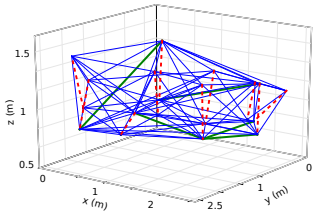

Fig. 2. The robot used for the locomotion experiment. The thick green lines indicate input springs, the dashed lines are bars (60 cm length, 5 cm diameter) and the thin lines are normal springs (output only).

We first briefly explain the setup and the training procedure, then the results and finally a short discussion. The robot consisted of 10 bars and 81 springs, of which 5 were actuated. This robot was placed on a flat terrain at the origin.

During training the desired output pattern (a non-linear oscillator such as the Van der Pol oscillator) is used instead of the output computed as a linear combination of the system's state. Hence, during training the desired output is fed back into the system. The states of the system are collected during the training phase (100 s) and used by the learning algorithm to modify the weights from the system's state to the output. To train the weights, we used linear regression.When the training phase ends, the testing phase (10000 s) starts and the actual output is fed back into the system.

Fig. 1 shows a result of our approach. The robot could maintain a forward velocity of about 0.4 km/h. This is not the maximum or energetically optimal velocity, but it shows that with even simple patterns gaits can be produced. We used the two-dimensional quadratic oscillator from [1] to generate the desired pattern for the results presented in Fig. 1, but we obtained similar results with other oscillators. While there is no guarantee that these patterns are useful for locomotion, they can serve as a basis for further optimization as there is already a certain oscillatory behavior in the system.

There is an important advantage of this approach: as the gait is produced by the combination of the body *and* the environment, one gets sensor integration for free. The robot only moves because of the combination of environment and body. The sensors only provide a static readout of the state of the body and it is this readout which is used to drive the motors. In this sense the body itself is the pattern generator as it is the state of the body that defines how the motors are driven. Indeed, if the body has rich enough dynamics, there is no need for an additional pattern generator which tracks the the system using sensor information.

An interesting phenomenon that we did not observe in preliminary experiments without gravity is that gaits become more robust when noise is injected through the feedback both during training and testing (more than the small amount necessary to avoid overfitting). One explanation for this is that multiple equilibrium configurations can exist and the robot will tend to fall into one of these when it is not actuated or not actuated enough to avoid being trapped in such a state. This

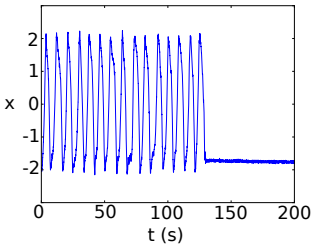

Fig. 3. One output variable of the system after learning a Van der Pol oscillator. After a while, the robot gets trapped in an equilibrium position. By adding noise to the feedback connections, this can prevented.

is a problem for linear feedback as once the robot is trapped in a stable state and the kinetic energy drops, it won't be able to get out of this configuration. The noise injects energy into the system to prevent these absorbing configuration from halting the system. For the locomotion experiment, only a small amount of noise was necessary (GWN with $\tau = 0.1$). Fig. 3 shows a similar experiment in which almost no noise was injected through the feedback. After about 130 s the robot got trapped in a stable state and couldn't get out anymore.

IV. Sensing

As the gaits developed by the robot inherently use sensor feedback, we may expect that the body can also extract environmental features. We verified this with a terrain classification task. We simulated two types of terrain consisting each of evenly spaced hemispheres placed on the ground as shown in Fig. 4. The terrains differ by the radius of the hemispheres, one has spheres with a radius of 7 cm, the other 15 cm. During training and testing, the terrain is switched every 100 s (one episode) and the robot is moved to a random position.

The robot used for this task consisted of 25 bars and 258 springs. This ensures that the robot has enough computational power to learn to maintain a gait on both terrains and to extract information from the environment. A larger robot was needed than for the locomotion experiment, because the same readout should work on both terrains, which is a harder task.

As in the locomotion experiment, the robot should first learn to maintain a stable oscillatory pattern. This time it has to maintain the same limit cycle on both types of terrain. We chose the Van der Pol oscillator as the target pattern. This is performed in the first training phase, lasting for 80 episodes.

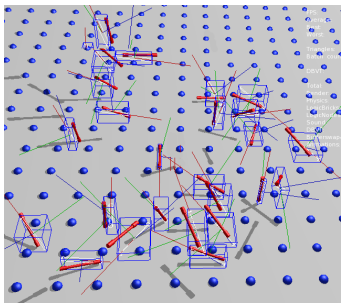

Fig. 4. The robot on one of the terrains used for the sensing experiment (the springs are not shown for clarity).

Afterwards, the robot generates the desired output pattern independently (as in the testing phase for the locomotion experiment) for another 80 episodes. The 40 first episodes of the free run phase are used to train the terrain classifier. The final 40 episodes are used to test the classifier. This means that there are 20 train and test episodes for each terrain.

Next, the classifier was trained using the 40 training episodes for the terrain classifier. Linear regression was used with the length of the springs as input variables and one output variable (1 for the first terrain, -1 for the other). Classification is then performed by averaging the output over one episode and applying the sign function. The error measure used was 0-1 loss. While linear regression is not an optimal training algorithm for classification, we used it for this experiment as it is a linear classifier (the robot provides the non-linearities) and the classic training technique for Reservoir Computing. Similar results were obtained with other linear classifiers.

The classifier output (before the sign function) on the classfier test set is shown in Fig. 5. The test error rate was 5/40 (12.5%), while the train error rate was 1/40 (2.5%).

For the sensing experiment, a larger amount of noise was applied, as the robot could more easily get trapped in a stable configuration (due to the terrain and increased weight). Adding more noise as the environments get more complex is not a viable general solution, as the robot will become noise driven.

The neural circuit in our approach (linear feedback) is one of the simplest imaginable and already provides some interesting behaviors. If one needs additional computational power that the body itself cannot provide (e.g. long-term memory), more complex neural circuits can be added. As long as one keeps the body in the loop and uses a bottom-up approach, the aforementioned advantages should allow for a robust solution.

V. Conclusion

We presented experimental results from a first exploration of a computational approach to morphological computation applied to tensegrity-like structures. We were able to produce stable locomotion patterns in these structures with only linear feedback. Next, the problem of extracting environmental information was addressed with a simple classification task.

The current approach was quite ad-hoc, because we imposed a desired output pattern and used random feedback connections. There is no guarantee that one will obtain optimal gaits with this technique. But because the system already generates oscillating patterns autonomously, this might be a good foundation for gait optimization. Gaits obtained in this way automatically use sensor feedback through the body itself. This is a more natural approach than starting with an independent pattern generator and then adding sensor information.

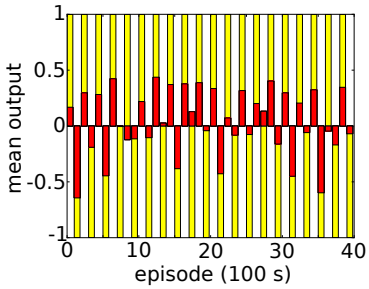

Fig. 5. Test output for the sensing experiment. The light bars are the desired output (1 for terrain 1, -1 for terrain 2), the dark bars are the system's output.

Where to go from here? One interesting path is to incorporate realistic muscle models [2] to verify if the computational power of the system increases. Tensegrity also seems to play a role at the cellular level [3], so one may ask on which level does morphological computation begin? Finally, how are body and mind related as they are both dynamical systems with computational power? Our experiments show that not much is needed to maintain a gait or to detect basic properties of the environment. So how do we develop powerful controllers for compliant robots by combining the computational aspects of neural networks and physical systems?

Acknowledgment

This research was funded by a Ph. D. fellowship of the Research Foundation - Flanders (FWO) and the European Community's Seventh Framework Programme FP7/2007-2013 Challenge 2 Cognitive Systems, Interaction, Robotics under grant agreement No 248311 - AMARSi.

References

[1] H. Hauser, R. Pfeifer, A. J. IJspeert, and W. Maass. A theoretical foundation for morphological computation. *[in preparation]*.

[2] A. V. Hill. The heat of shortening and the dynamic constants of muscle. *Proceedings of the Royal Society of London Series B Biological Sciences*, 126(843):136–195, 1938.

[3] D. E. Ingber. Tensegrity: the architectural basis of cellular mechanotransduction. *Annual Review of Physiology*, 59(1):575–599, 1997.

[4] W. Maass, P. Joshi, and E. D. Sontag. Computational Aspects of Feedback in Neural Circuits. *PLoS Computational Biology*, 3(1):20, 2007.

[5] C. Paul, J. W. Roberts, H. Lipson, and F. J. Valero Cuevas. Gait production in a tensegrity based robot. *ICAR 05 Proceedings 12th International Conference on Advanced Robotics 2005*, pages 216–222, 2005.

[6] C. Paul, H. Lipson, and F. J. V. Cuevas. Evolutionary form-finding of tensegrity structures. *Proceedings of the 2005 conference on Genetic and evolutionary computation GECCO 05*, page 3, 2005.

[7] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. 2007 special issue: An experimental unification of reservoir computing methods. *Neural Networks*, 20(3):391–403, 2007.