Locomotion Without a Brain:

Physical Reservoir Computing in Tensegrity Structures

K. Caluwaerts (Ghent Univ.) *et al.*
Artificial Life 2013

Presented by Bongjun Kim

2021-11-14

Computer Graphics @ Korea University

KOREA UNIVERSITY

KUCG

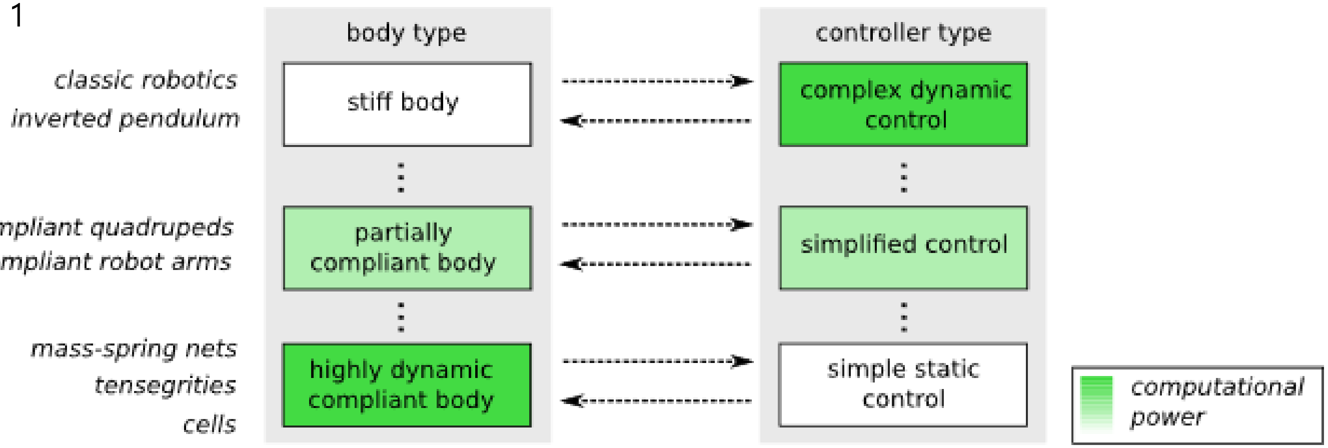
Index

- Introduction
- Related Work
- Tensegrity Structure
- Central Pattern Generators (CPG)
- Physical Reservoir Computing
- Outsourcing Motor Pattern Generation
- Experiment Result
- Discussion and Conclusion

Introduction

Introduction
Robotics

Fig. 1

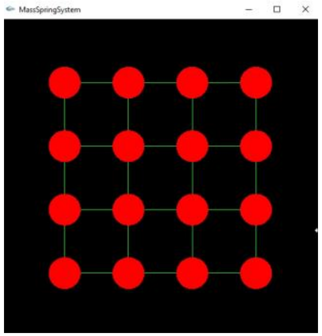


https://www.thingbits.net/

stiff body
(classic robotics)

https://www.youtube.com/

partially compliant body
(compliant quadrupeds)

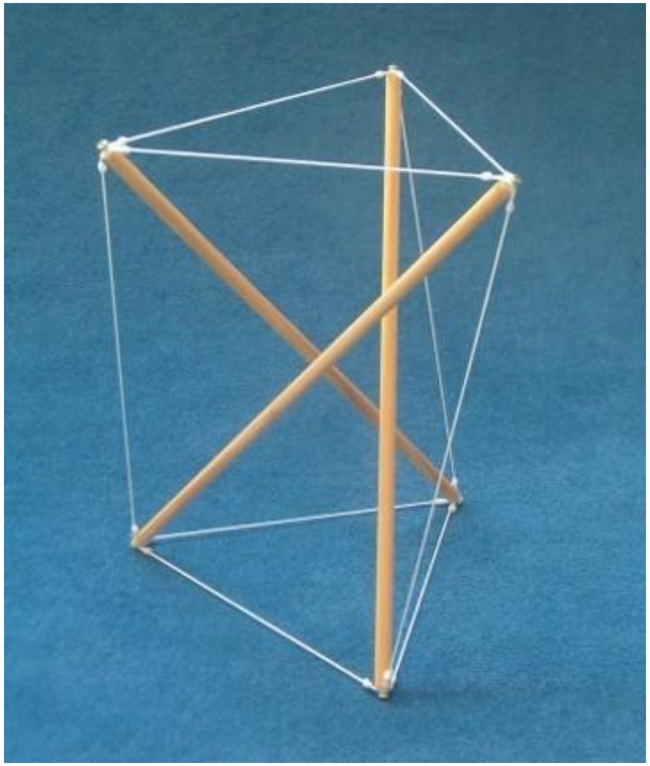


highly dynamic
compliant body
(mass-spring nets)

Introduction
Tensegrity Structure

- Structure Components
  - String
    - Only resisting tensile forces, not compression
    - Mass properties neglected.
  - Bar
    - Resisting compressive and tensile force
    - Assumed infinitely thin.
    - Inertia moment exist only perpendicular to the longitudinal axis.

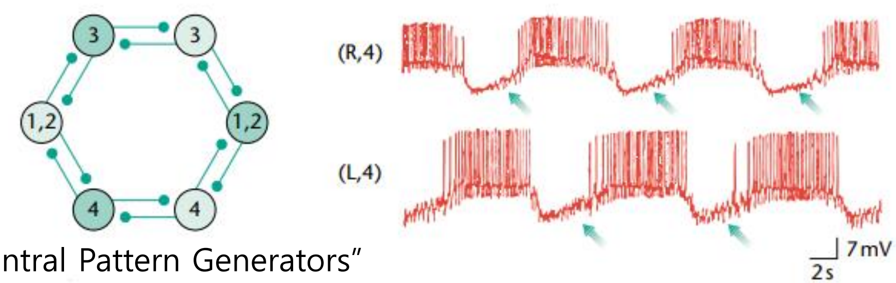
- Benefits
  - High Strength to Weight Structure
  - Passive Global Force Distribution
  - Minimized Points of Local Weakness



http://www.tensegriteit.nl/e-simple.html

Introduction
Central Pattern Generators (CPG)

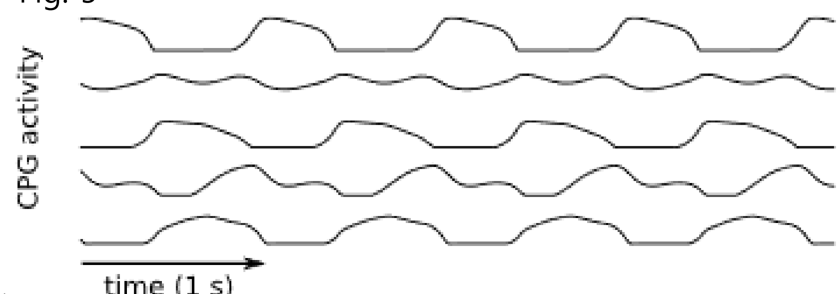
- Biological neural circuits, typically found in the spine of vertebrates.
- Generate rhythmic activation patterns, source of the rhythmic motions. (e.g. walking, breathing)
- Sustained activity without sensory feedback or higher-level control input.



"Central Pattern Generators"
[Scott Hooper (Ohio Univ.) *et al.* / Encyclopedia of Life Sciences, 2001]

(Leech heartbeat rhythm-generating network)

Fig. 5

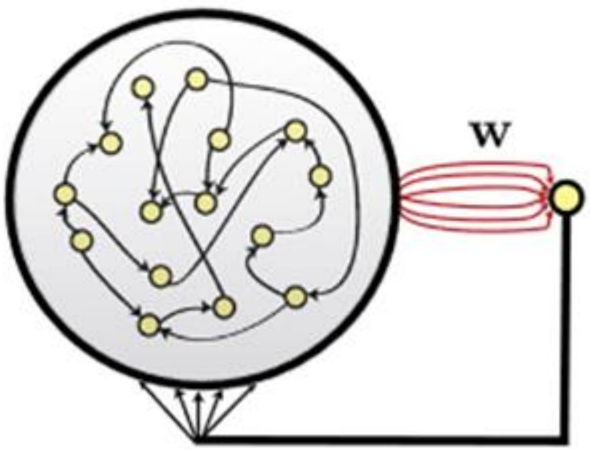


(Rate based CPG signal model)

Introduction
Reservoir Computing

- Modify only readout weights, much simpler to train such recurrent neural networks.

- Reservoir computing, originally applied only to neural networks, has since been extended to other nonlinear dynamical systems, called PRC.



"Generating Coherent Patterns of Activity from Chaotic Neural Networks"

[David Sussillo (Columbia Univ..) and L.F. Abbott / Neuron, 2009]

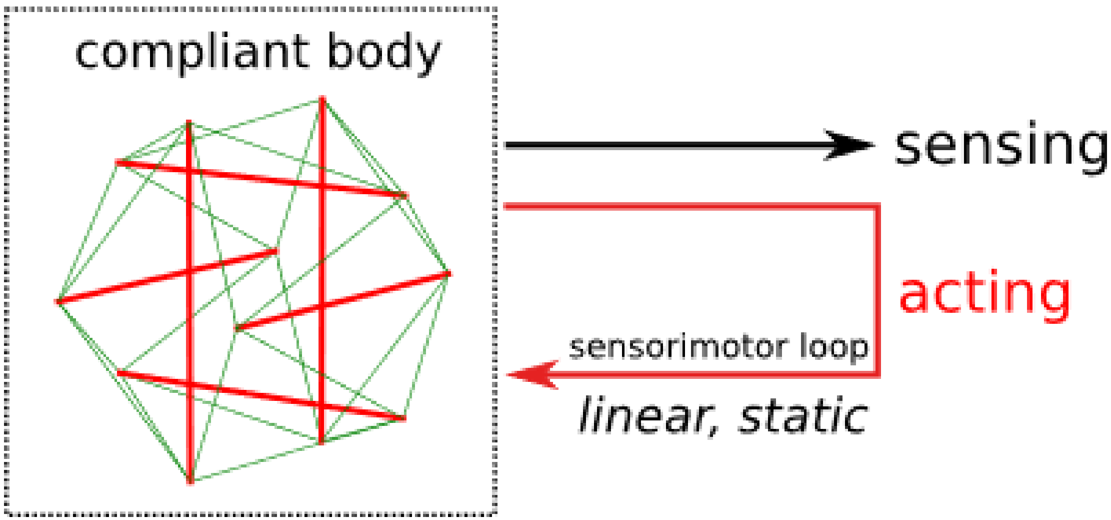
Abstract

- Studying tensegrity structure, one of highly dynamic body structures.

- Show tensegrity structures can maintain complex gaits.
  - Demonstrate the existence of a spectrum of choices of how to implement control in the body-brain composition.

Contribution

- Show compliant robots have real computational power.
- Using tensegrity structures, provide an implementation of the general principle of PRC.



Over View

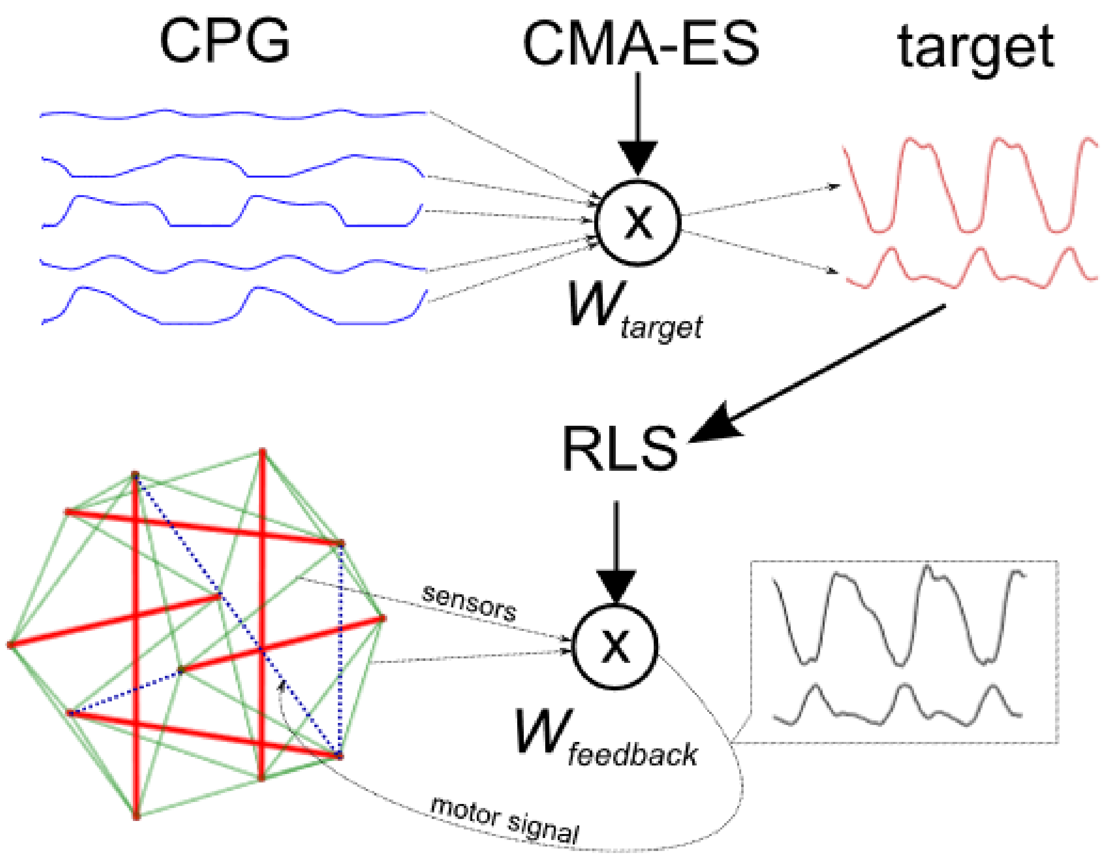
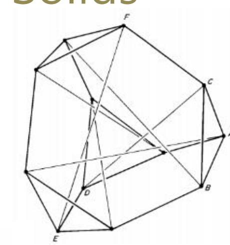


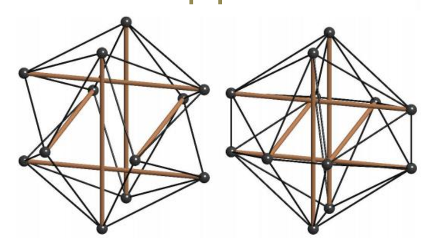
Fig. 15

Related Work

Related Work
Tensegrity Structure

- Investigating equilibrium and stiffness of frames.
  - "Buckminster Fuller's tensegrity structures and Clerk Maxwell's rules for the construction of stiff frames"
    - [C. R. Calladine (Cambridge Univ.) / International Journal of Solids and Structures, 1978]
  - "The stiffness of tensegrity structures"
    - [S. D. Guest (Cambridge Univ.) / IMA Journal of Applied Mathematics, 2011]

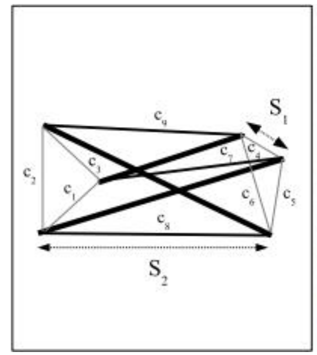




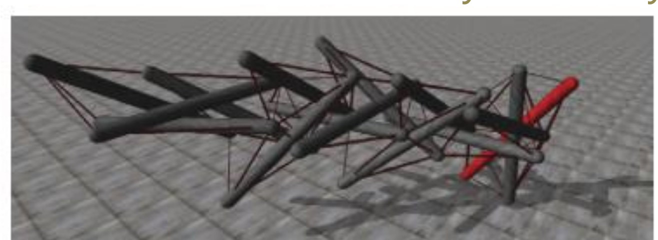
- Investigated the linearized dynamics.
  - "Linear dynamics of tensegrity structures"
    - [Cornel Sultan (Harvard Univ.) *et al.* / Engineering Structures, 2002]

Related Work
Tensegrity Robot Control

- Gait obtained by genetic algorithm, through actuating black springs.
  - "Gait Production in a Tensegrity Based Robot"
    - [Chandana Paul (Cornell Univ.) *et al.* / International Conference on Advanced Robotics, 2005]




- ANN, 15 bar, 30 sensor and actuator at each end of each bar
  - "Morphological communication: exploiting coupled dynamics in a complex mechanical structure to achieve locomotion"
    - [John A. Rieffel (Cornell Univ.) *et al.* / Journal of the Royal Society Interface, 2010]

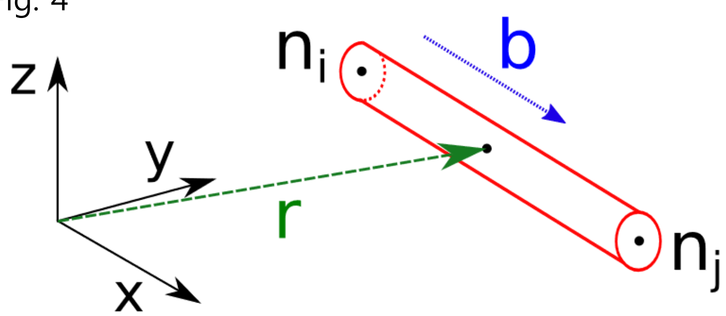


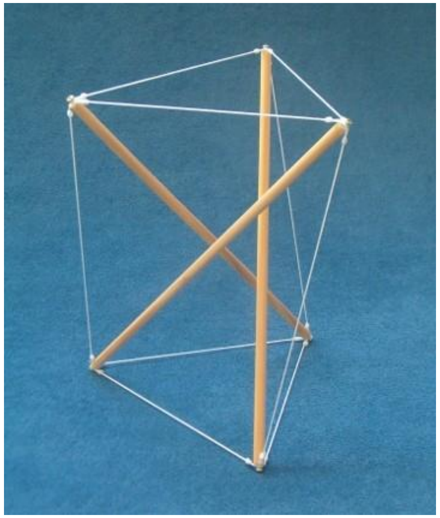
Tensegrity Structure

Tensegrity Structure
Dynamics (1/2)

- $n = 2b$ (n is number of nodes, b is number of bars)
- $\boldsymbol{r}$ is fixed to the center of mass of the bar
- $\boldsymbol{b}$ is a unit vector along the longitudinal axis of the bar
- $\mathbf{N} = \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & y_n \\ z_1 & \cdots & z_n \end{bmatrix} = \boldsymbol{Q}\boldsymbol{\Psi}^T$ (nodes position in cartesian coordinates)
- $\boldsymbol{Q} = [\boldsymbol{r}_1 \quad \cdots \quad \boldsymbol{r}_b \quad \boldsymbol{b}_1 \quad \cdots \quad \boldsymbol{b}_b], \boldsymbol{\Psi} = \begin{bmatrix} \mathbf{I} & \mathbf{L}/2 \\ \mathbf{I} & -\mathbf{L}/2 \end{bmatrix}$
  - $\mathbf{L}$ contains the lenghts of the bars

Fig. 4



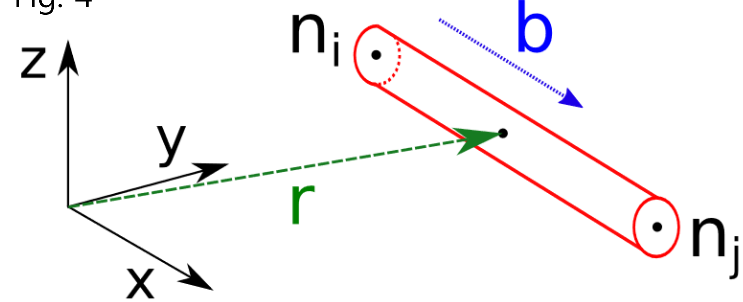


http://www.tensegriteit.nl/e-simple.html

Tensegrity Structure
Dynamics (2/2)

- $\mathbf{F}_q = (\mathbf{W} - \mathbf{N}\mathbf{C}^T diag(\lambda)\mathbf{C})\boldsymbol{\Psi} = (\mathbf{W} - \boldsymbol{Q}\boldsymbol{\Psi}^T\mathbf{C}^T diag(\lambda)\mathbf{C})\boldsymbol{\Psi}$
- $\mathbf{W}$ contains the external forces acting on the nodes
  - $\mathbf{C} \in \{0, 1, -1\}^{s \times n}$ called the connectivity matrix
    - s is number of springs at least $\frac{3n}{2}$ (each node is connected to at least three springs)
  - $\lambda = \max\left(k\left(1 - \frac{l_0}{\|\boldsymbol{n}_i - \boldsymbol{n}_j\|}\right), 0\right)$
  - $\mathbf{W} = \mathbf{W}_{ext} + \dot{\mathbf{N}}\mathbf{R}$
    - damping along the springs $\mathbf{R} = \zeta\mathbf{C}^T\mathbf{C}$
    - uniform damping coefficient ζ

Fig. 4



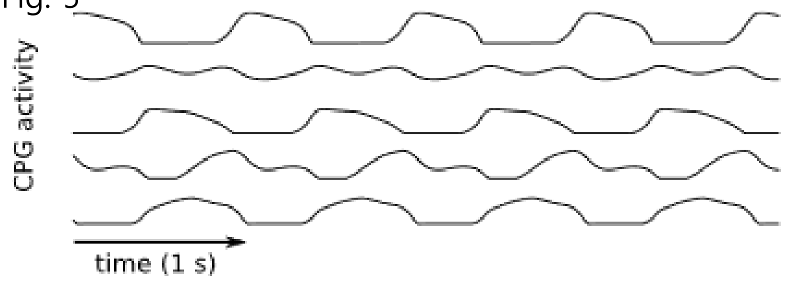
- $\mathbf{F}_q = \mathbf{M}(\ddot{\boldsymbol{Q}} + \boldsymbol{\Xi}\boldsymbol{Q})$
  - mass matrix $\mathbf{M} = diag([m_1 \quad \cdots \quad m_b \quad j_1 \quad \cdots \quad j_b])$
    (j_i are the moments of inertia of the bars)
  - describing the rotational equation $\boldsymbol{\Xi} = diag([0 \quad \cdots \quad 0 \quad \xi_1 \quad \cdots \quad \xi_b])$
    - ***lagrange multipliers*** $\xi_i = \frac{\dot{\boldsymbol{b}}_i^T\dot{\boldsymbol{b}}_i + j_i^{-1}\boldsymbol{b}_i^T\mathbf{F}_{q,(b+i)}}{\boldsymbol{b}_i^T\boldsymbol{b}_i}$ ($\mathbf{F}_{q,(b+i)}$ is $b + i$th column of $\mathbf{F}_q$)

Central Pattern Generator

Central Pattern Generators (CPG)
Matsuoka Oscillators (1/2)

- $\dot{\boldsymbol{x}}^{osc} = \frac{-\boldsymbol{x}^{osc} - \mathbf{A}\boldsymbol{y}^{osc} + \boldsymbol{\gamma} - \boldsymbol{\iota}\boldsymbol{v}^{osc}}{\tau_1}, \dot{\boldsymbol{v}}^{osc} = \frac{\boldsymbol{y}^{osc} - \boldsymbol{v}^{osc}}{\tau_2}, \boldsymbol{y}^{osc} = \max(\boldsymbol{x}, \mathbf{0})$
  - $\boldsymbol{x}^{osc}$ = signal of neuron
  - $\boldsymbol{y}^{osc}$ = output of the oscillator
  - $\boldsymbol{v}^{osc}$ = term of models fatigue (generate relaxation)
  - $\mathbf{A}$ = matrix, how the neurons are connected
  - $\tau_1 = 0.5, \tau_2 = 5$ (time constants)
  - $\boldsymbol{\iota} = 1$ (the steady state firing rate of the neuron)
  - $\boldsymbol{\gamma} = 1$ (impulse rate of the tonic or slowly varing input)

Fig. 5

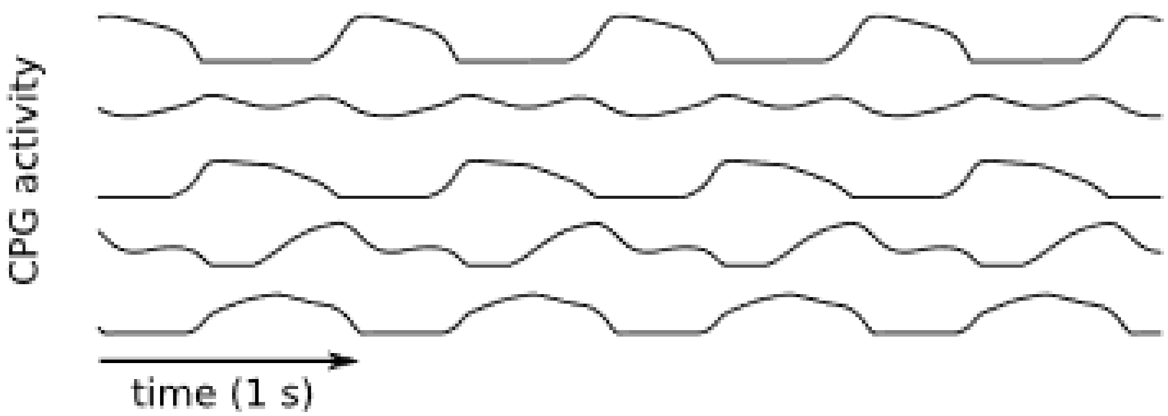


Central Pattern Generators (CPG)
Matsuoka Oscillators (2/2)

- $\boldsymbol{b} \in [0,1]^n,\ \mathbf{A} = \mathbf{C}^T diag(\boldsymbol{b})\mathbf{C} - diag(diag(\mathbf{C}^T diag(\boldsymbol{b})\mathbf{C}))$
  - $\mathbf{C}$ = connectivity matrix of spring

- $\boldsymbol{y}^{target} = \mathbf{W}^{target}\boldsymbol{y}^{osc}$
  - $\boldsymbol{y}^{target}$ = target motor signal
  - $\mathbf{W}^{target}$ = bias matrix for CPG signal to motor signal

Fig. 5



Physical Reservoir Computing

Physical Reservoir Computing
Dynamics

- $\boldsymbol{x}[d+1] = \tanh(\mathbf{W}_{res}\boldsymbol{x}[d] + \mathbf{W}_{in}\boldsymbol{u}[d+1])$
- $\boldsymbol{y}[d+1] = \mathbf{W}_{out}\boldsymbol{x}[d+1]$
  - $\mathbf{W}_{out}$ = readout matrix
  - $\mathbf{W}_{res}$ = fade out(based on the spectral radius) and connectivity matrix
  - $\mathbf{W}_{in}$ = input weight matrix

- Implement functions that fading memory property, not necessary
  - $\boldsymbol{x}[d+1] = \tanh\left(\mathbf{W}_{res}\boldsymbol{x}[d] + \mathbf{W}_{in}\boldsymbol{u}[d+1] + \mathbf{W}_{fb}\boldsymbol{y}[d]\right)$
  - $\boldsymbol{y}[d+1] = \mathbf{W}_{out}\boldsymbol{x}[d+1]$
    - $\mathbf{W}_{fb}$ = feedback weight matrix, typically chosen at random

Apply to Tensegrity Structure (1/2)

- system state defined $\boldsymbol{x}(t) = vec\begin{pmatrix} \boldsymbol{f}(t) & \dot{\boldsymbol{f}}(t) \\ \boldsymbol{f}(t-\Delta) & \dot{\boldsymbol{f}}(t-\Delta) \\ \vdots & \vdots \\ \boldsymbol{f}(t-k\Delta) & \dot{\boldsymbol{f}}(t-k\Delta) \end{pmatrix}$
  - Δ= time step
  - k = the number of delay steps
  - spring forces measured at time t, $\boldsymbol{f}(t)$
    - can be written as $f_e(t) = \max\left(k_e\left(\left\|\boldsymbol{n}_i - \boldsymbol{n}_j\right\| - l_{0,e}(t)\right), 0\right)$
  - the equilibrium lenghts explicitly use the time index $l_{0,e}(t)$

Apply to Tensegrity Structure (2/2)

- $\boldsymbol{l}_0^{act}(t) = l_{max}g\big(\boldsymbol{y}(t)\big) + \boldsymbol{l}_0^{act}(0)$
  - $\boldsymbol{l}_0^{act}$ = subset of actuated spring, $\boldsymbol{l}_0^{pas}$ = subset of passive spring
  - l_{max} = maximum change of springs about equilibrium length
  - $g : \mathbb{R}^a \rightarrow [-1, 1]^a$
    - a = the number of actuated springs

- $\boldsymbol{y}(t) = \mathbf{W}\boldsymbol{x}(t)$
  - $\boldsymbol{y}(t)$ = signal to motor about length
  - $\mathbf{W}$ = constant bias input (study to optimized)
  - $\boldsymbol{x}(t)$ = force sensor measurements from the tensegrity structure

Outsourcing Motor Pattern Generation

Outsourcing Motor Pattern Generation
Recursive Least-Squares Approach (1/2)

- Training algorithm to learn target signal.
- By RLS, same samples compute the same weights as batch linear regression.

- Advantage of RLS
  - Gradually transition from a completely teacher-forced structure(the desired signals are fed into the system) to a system generating its own control signals.

- Disadvantage of RLS
  - Needs to update the covariances matrix of all the input variables, does not scale well
  - Have to know explicit target signal

Outsourcing Motor Pattern Generation

Recursive Least-Squares Approach (2/2)

- $y_i(t) = \alpha_{rls} y_i^{target}(t) + (1 - \alpha_{rls}) \sum_j \mathrm{W}_{i,j}^{rls}(t) x_j(t)$
  - $\alpha_{rls} = \frac{1}{1+\tau_{rls} t} \quad \text{if } t < \text{train time}, \quad \text{else } 0$
    - $\tau_{rls} = \text{teacher} - \text{forcing decay time constant}$
  - $\mathbf{W}^{rls} = \text{weights, updated at eachtime step by RLS equations}$
  - $\mathbf{W}^{rls}(t) = \mathbf{W}^{rls}(t - \Delta t) + \mathbf{L}^{rls}(t) \boldsymbol{e}^{rls}(t)$
    - $\mathbf{L}^{rls}(t) = \frac{\mathbf{P}^{rls}(t)\boldsymbol{x}(t)}{1+\boldsymbol{x}^T(t)\mathbf{P}^{rls}(t)\boldsymbol{x}(t)}$
      - $\mathbf{P}^{rls} = \text{covariance matrix, initialized to identity matrix}$
      - $\mathbf{P}^{rls}(t + \Delta t) = \mathbf{P}^{rls}(t) - \frac{\mathbf{P}^{rls}(t)\boldsymbol{x}(t)\boldsymbol{x}^T(t)\mathbf{P}^{rls}(t)}{1+\boldsymbol{x}^T(t)\mathbf{P}^{rls}(t)\boldsymbol{x}(t)}$
    - $\boldsymbol{e}^{rls}(t) = \boldsymbol{y}^{target}(t) - \mathbf{W}^{rls}(t - \Delta t)\boldsymbol{x}(t)$

Outsourcing Motor Pattern Generation
Gradient Descent Approach

- To overcome the disadvantage of the RLS algorithm, needs to update the covariances matrix of all the input variables each time steps.

- Gradient descent rule converges slower in practice than the RLS rule, because of the learning rate α^{gd} has to be chosen small enough to prevent instability.

- $\mathbf{W}^{gd}(t) = \mathbf{W}^{gd}(t - \Delta t) - \alpha^{gd}\boldsymbol{e}^{rls}(t)\boldsymbol{x}^{T}(t)$
  - Equation is obtained easily by differentiating the quadratic error at a time step, replace the update of $\mathbf{W}^{rls}$

Reward-Modulated Hebbian Approach(1/2)

- Use reward signal instead of error signal
- Reward signal is large when the error is small, and vice versa.

- Hebbian rule
  - $\Delta \mathrm{W} \propto y(\text{output})x(\text{input})$

- The reward-modulated Hebbian rule(RMH) we used is given by
- $\boldsymbol{y}(t) = \mathbf{W}^{rmh}(t - \Delta t)\boldsymbol{x}(t) + \boldsymbol{v}(t))$
- $\mathbf{W}^{rmh}(t) = \mathbf{W}^{rmh}(t - \Delta t) - \alpha^{rmh}\boldsymbol{v}(t)\big(R(t) - \bar{R}(t)\big)\boldsymbol{x}^{T}(t)$
  - $R(t) = \text{reward signal}$
  - $\bar{R}(t) = \text{average of reward signal during the last 100 ms}$

Outsourcing Motor Pattern Generation
Reward-Modulated Hebbian Approach(2/2)

- $\mathbf{W}^{rmh}(t) = \mathbf{W}^{rmh}(t-\Delta t) - \alpha^{rmh}(\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t))\big(R(t) - \bar{R}(t)\big)\boldsymbol{x}^T(t)$
  - $\boldsymbol{y}(t) - \bar{\boldsymbol{y}}(t)$=approximated $\boldsymbol{v}(t)$, when $\boldsymbol{y}(t)$ varies smoothly.
  - $\bar{\boldsymbol{y}}(t) = \text{average of reward signal during the last 100 ms}$
  - $R(t) = -\sum_i \left|y_i(t) - y_i^{target}(t)\right|$

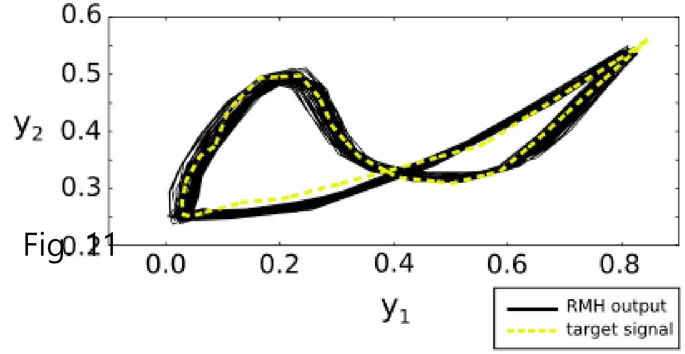


Fig. 11

- $\mathbf{W}^{rmh}(t) = \mathbf{W}^{rmh}(t-\Delta t) - \alpha^{rmh}\big(\boldsymbol{y}(t)$ ·
  - $R(t) = -\max_i \left|y_i(t) - y_i^{target}(t)\right|^2$

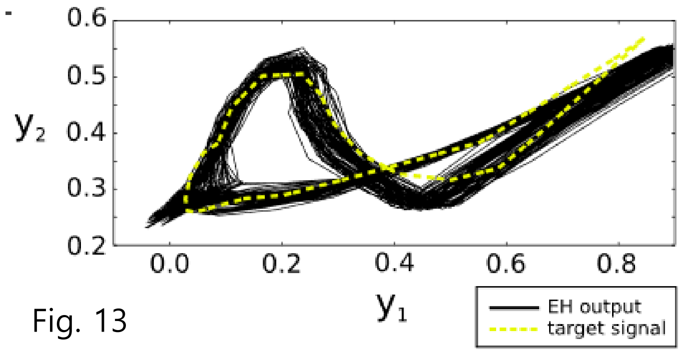


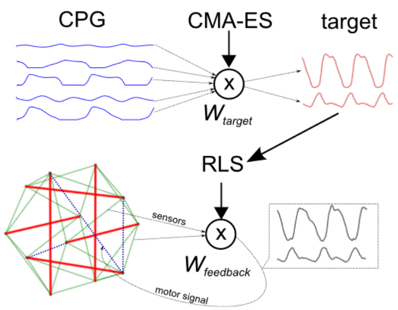
Fig. 13

Experiment Result

Experiment Result
Gait Optimization (1/3)

- Step
  - 1. Optimize $\mathbf{W}_{target}$, for a given basic CPG, by CMA-ES, to generate target signal.
    - CMA-ES
      - population size = 50
      - iterations time = 10
      - evaluation period = 30 s
        - only 4 h exploration time needed on real robot.
      - for four actuators
  - 2. Optimize $\mathbf{W}_{rls}$, for sensor input signal, by RLS, to generate motor signal.

Fig. 15



Experiment Result
Gait Optimization (2/3)

Fig. 16

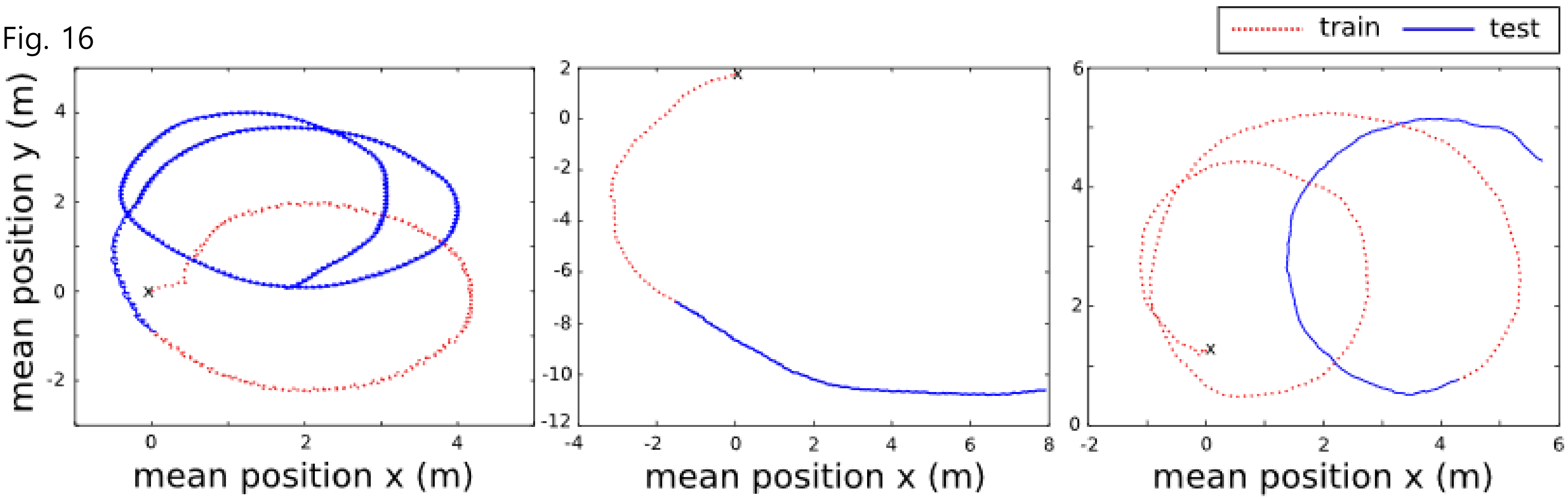


Fig. 2

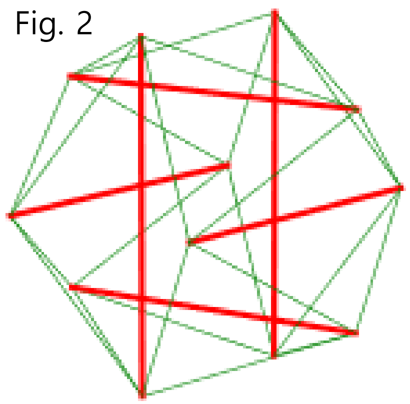


Fig. 17

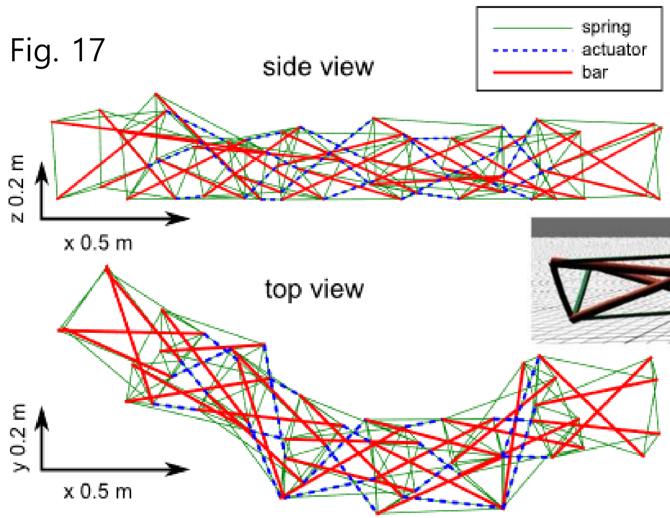
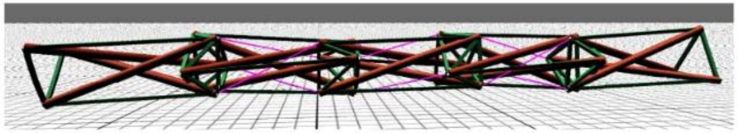


Fig. 3



Experiment Result
Gait Optimization (3/3)

- Result of ground reaction forces on the endpoints.

Fig. 18

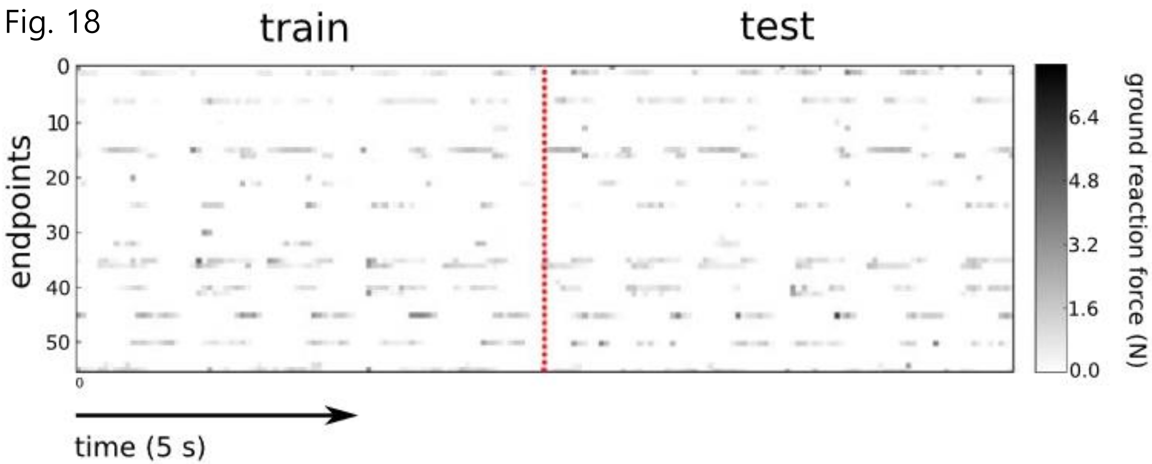
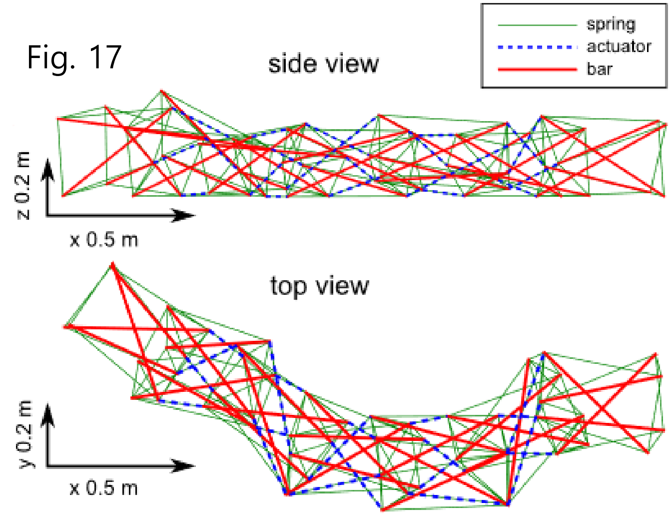


Fig. 17



- The sample is taken from beginning of the training(almost teacher forced) and end of testing(free run).

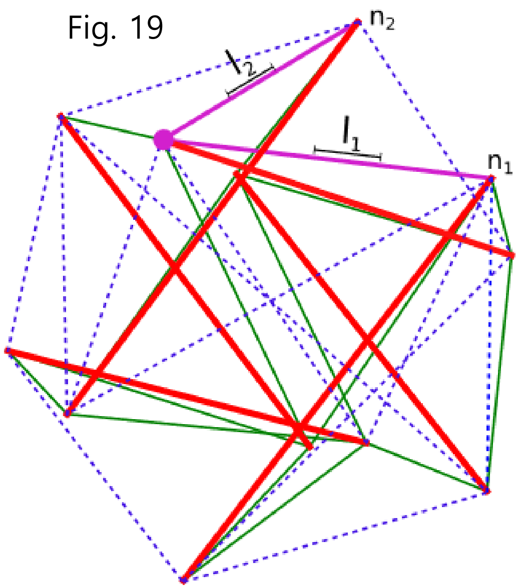
- The relative phase between the ground contacts is identical during training and testing.

Experiment Result
End-Effector Control

- Apply same technique to gait optimization.

- Simulation condition
  - Used 30-dimensional CPG, based on a connection pattern from a stacked tensegrity prism.
  - Simulated the system for 100 s
  - Mean Squared error over the last 80 s
  - RLS training in 75 s

Fig. 19



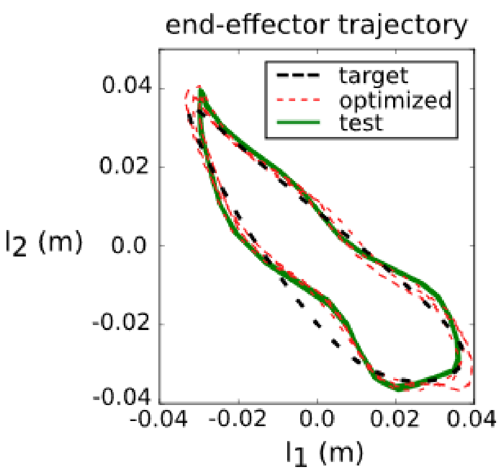


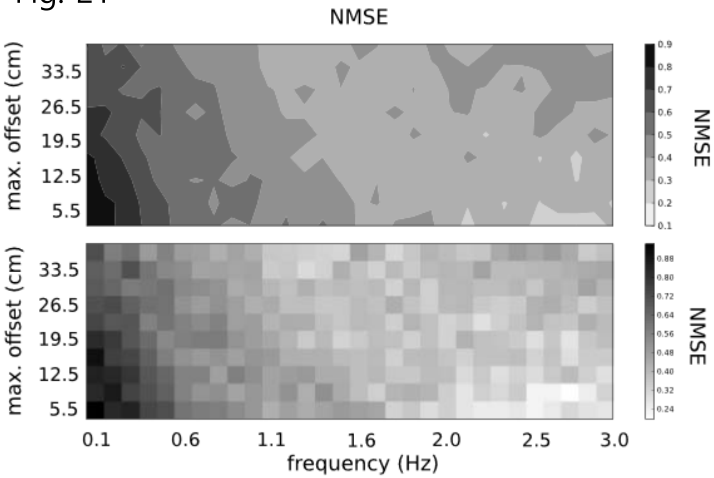
Figure 20. Trajectory of the end effector during testing after 75 s of training.

Experiment Result

The Importance of Complex Dynamics

- $\text{NMSE} = \frac{(\boldsymbol{x}-\boldsymbol{y})^T(\boldsymbol{x}-\boldsymbol{y})}{N\sigma(\boldsymbol{y})}$
  - NMSE=normalized mean squared error
  - N=the number of samples
  - $\boldsymbol{x}$=the vectorized output
  - $\boldsymbol{y}$=the vectorized target signal
  - $\sigma(\boldsymbol{y})$=activity function

Fig. 21



- Better performance obtained by increasing the frequency or the maximum spring equilibrium offset.
- Lager deformations of the structure cause the error to decrease.

Discussion and Conclusion

Discussions

- Compliance offers multiple advantages over classic, stiff robotics.
  - Safer robot-human interactions
  - Increased energy efficiency
  - Robustness against external perturbations
  - Simplification of the control

- Remarkable points
  - Do not need exact dynamics of the system to learning algorithm.
  - Historically tensegrity structures have been used to model a plethora of complex systems.

- Understanding cognition in biological organisms.
  - Allow to quantify the nature of the computation
  - Applicable to many of the interactions between the body, sensory inputs, but probably not enough to capabilities of human-level intelligence.

- The idea, principle of PRC, any size of system performs the same amount of computation, simply realizing different external perturbations.

Conclusions

- Simple linear learning rules to be able to learn complex locomotion patterns or desired end-effector trajectories.

- This provides a number of advantages from a robotic standpoint
  - The control complexity can be highly reduced
  - Very uninformative reward signals can be used to train complex pattern generators
  - The learned control law is robust to perturbations and can easily synchronize with environmental interactions.

- In conceptual point of view, the conclusion are more profound.
  - "dynamic bodies" only require "extremely simple brains" to implement computations
  - Opened up a whole spectrum of potential tradeoffs, between brain-based computation and body-based computation.
  - The results from the field of reservoir computing, implemented by the physical body, can be quantified