ETH zürich

Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation

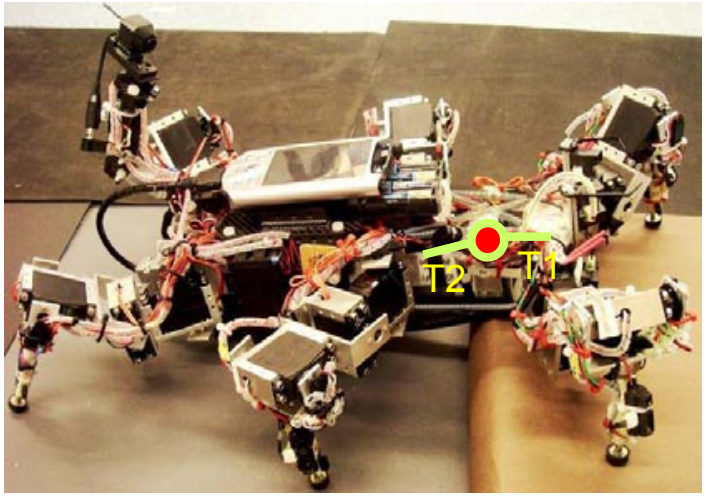
G. Ren, W. Chen, S. Dasgupta, Ch. Kolodziejski, F. Wörgötter, P. Manoonpong

Overview

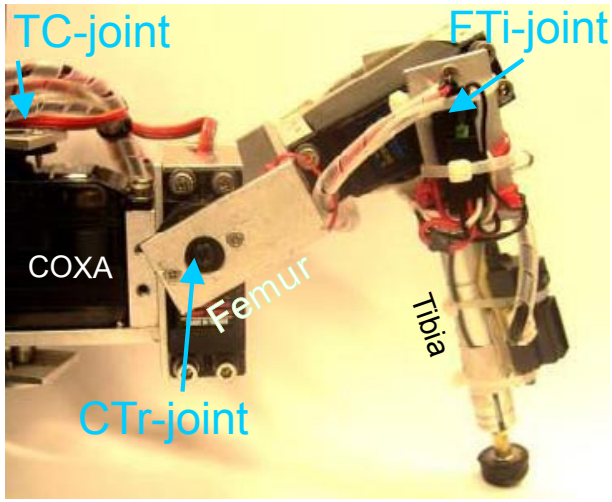
- A chaotic system is implemented to drive a legged robot's locomotion as a central pattern generator (CPG).
- From this setting, sophisticated gait patterns arise so that the robot can perform various walking behaviors.
- However, a single chaotic CPG controller has difficulties dealing with leg malfunction.
- They extended the single chaotic CPG to multiple CPGs with learning based on a simulated annealing algorithm.
- The multiple CPGs are synchronized to show identical dynamics, but with malfunction, the CPGs desynchronize, leading to independent leg dynamics.
- The learning mechanism automatically adjusts the remaining leg periodicities so that the robot can deal with the malfunction.

Meet AMOS II

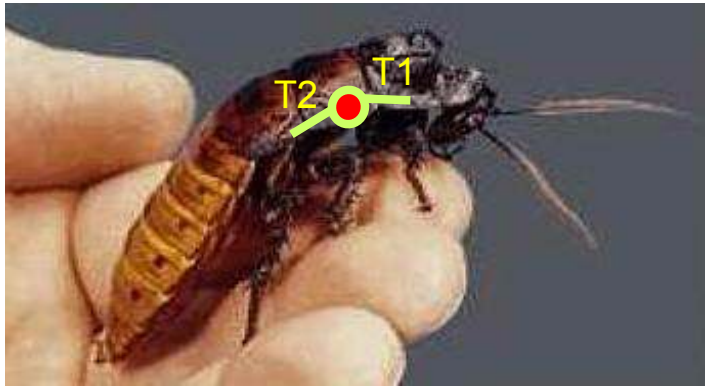
A



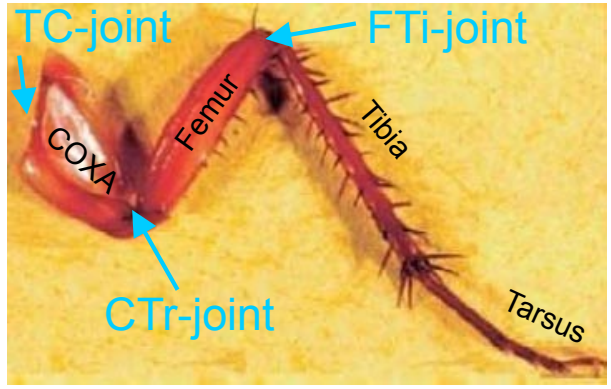
B



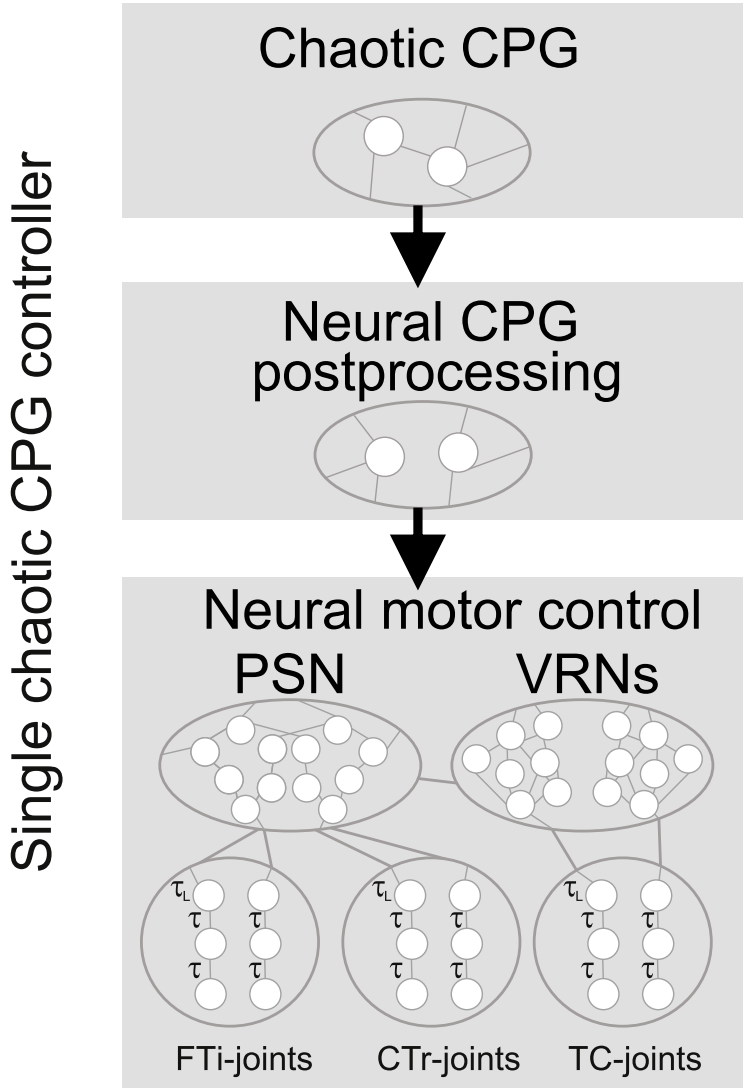
C



D



[1] From [2] S.Steingrube, M. Timme et al., Nature Physics 6 (2010)



- Adaptive neural chaos control (CPG)
  - Discrete time dynamics
  - Sigmoid activation function
  - Additional control signal *c* based on period p applied every $p + 1$ time step
- CPG post-processing
  - time window
  - hysteresis
  - signal integration
- Neural motor control
  - Phase switching network (PSN)
  - Velocity regulation network (VRN)
- Delay lines

Chaotic CPG overview

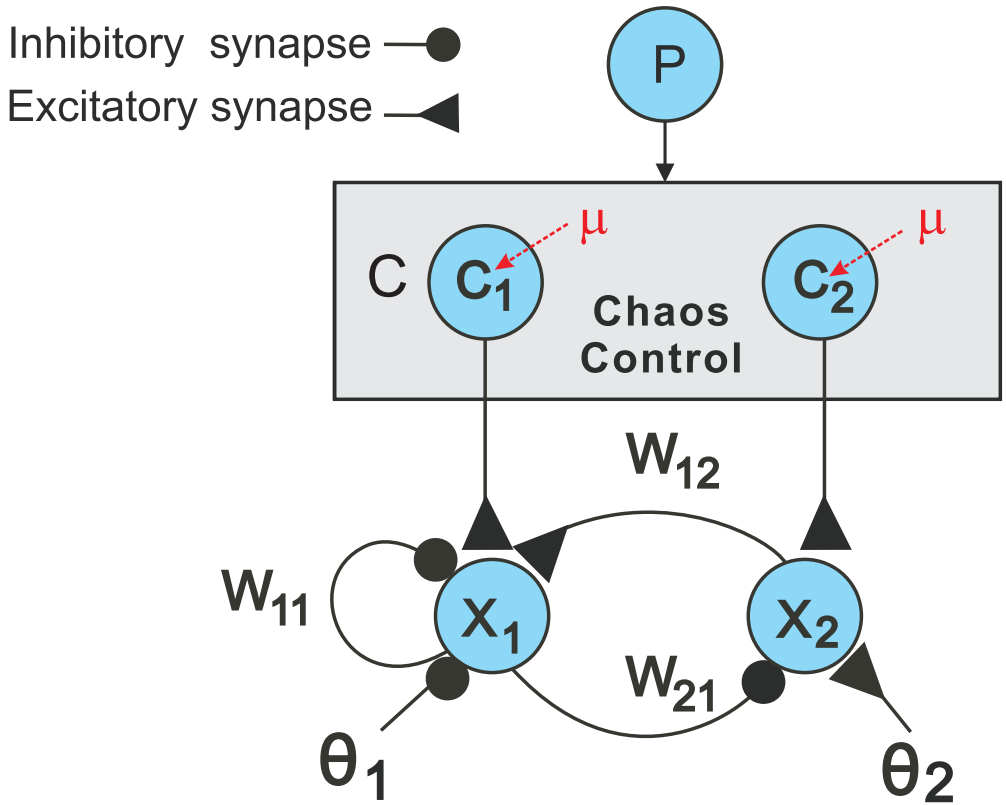


Fig. 1. Single CPG with the chaos controller.

[2] From [2] S.Steingrube, M. Timme et al., Nature Physics 6 (2010)

CPG activity (discrete time dynamics)

Neuron state

$$x_i(t+1) = \sigma\left(\theta_i + \sum_{j=1}^{2} w_{ij} x_j(t) + c_i^{(p)(t)}\right) \quad \text{for } i \in \{1,2\}$$

Control signal

$$c_i^{(p)(t)} = \mu_{(p)}(t) \sum_{j=1}^{2} w_{ij} \Delta_j(t)$$

$$\Delta_j(t) = x_j(t) - x_j(t-p)$$

$$\mu_{(p)}(t) = \mu_{(p)}(t) + \lambda \frac{\Delta_1^2(t) + \Delta_2^2(t)}{p} \quad \text{with } \lambda = 0.05$$

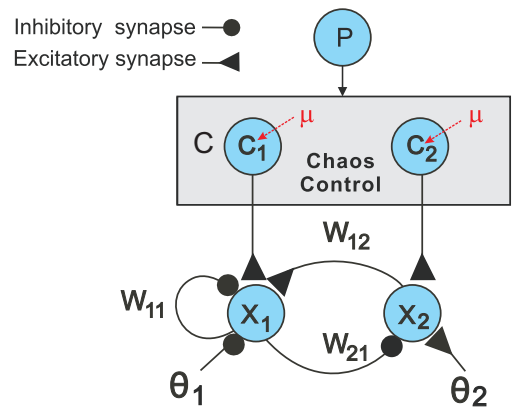


Fig. 1. Single CPG with the chaos controller.

Different hexapod gaits

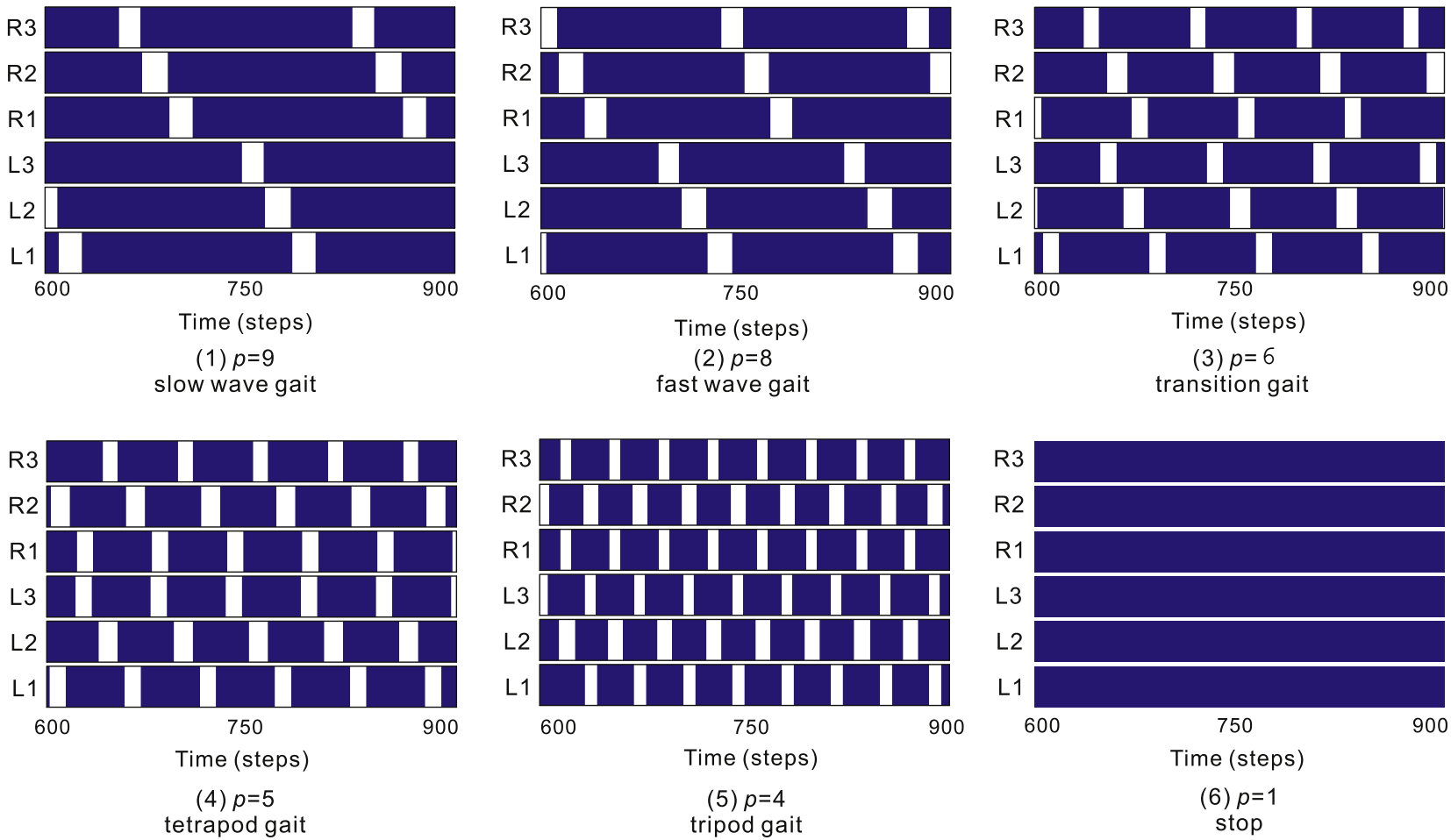


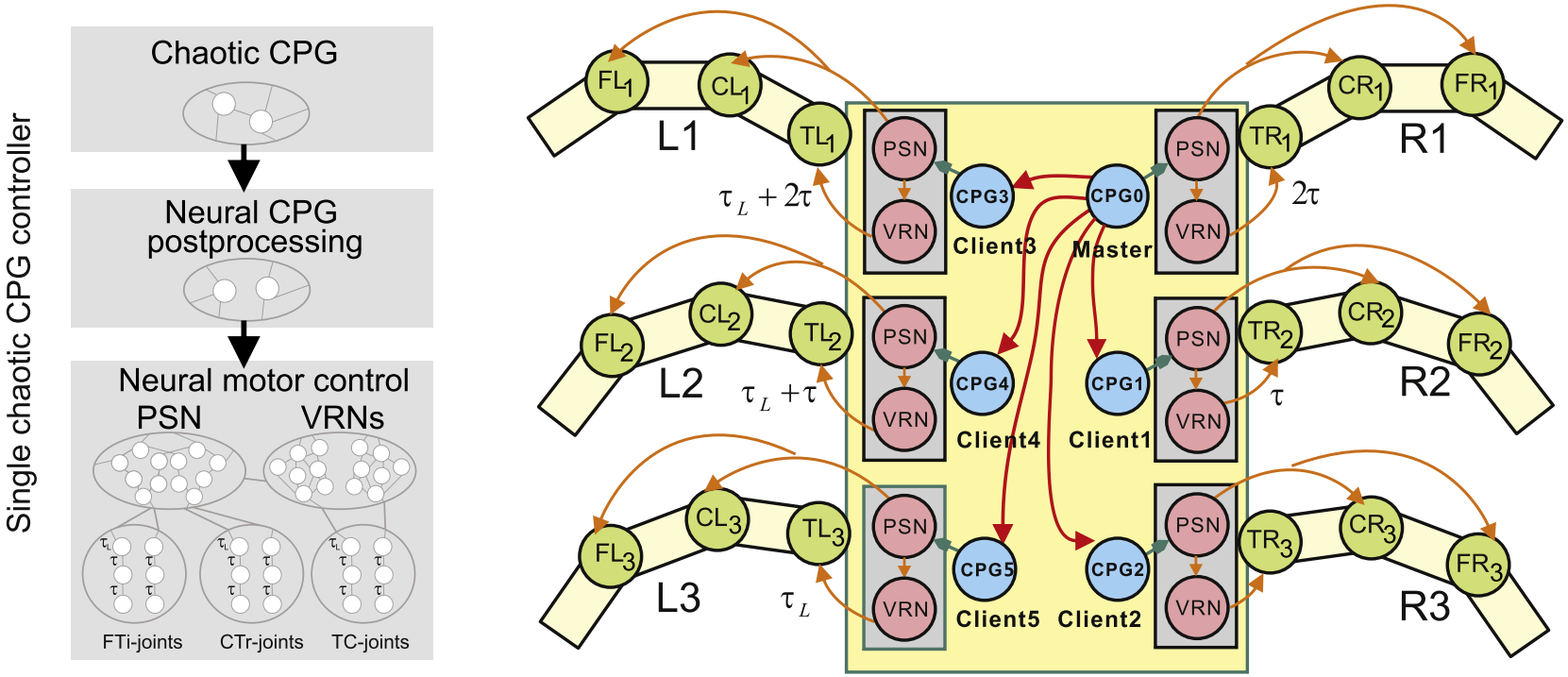
Fig. 2. Different hexapod gaits for changing p and the stop status ($p = 1$).

[3] From [2] S.Steingrube, M. Timme et al., Nature Physics 6 (2010)

Different gaits

[4] From [2] S.Steingrube, M. Timme et al., Nature Physics 6 (2010)

Multi CPG extension



(a) Single CPG controller (b) Multiple CPGs controller

Fig. 3. Single chaotic CPG (a) and multiple chaotic CPGs (b) for a multi-legged robot.

Client CPG

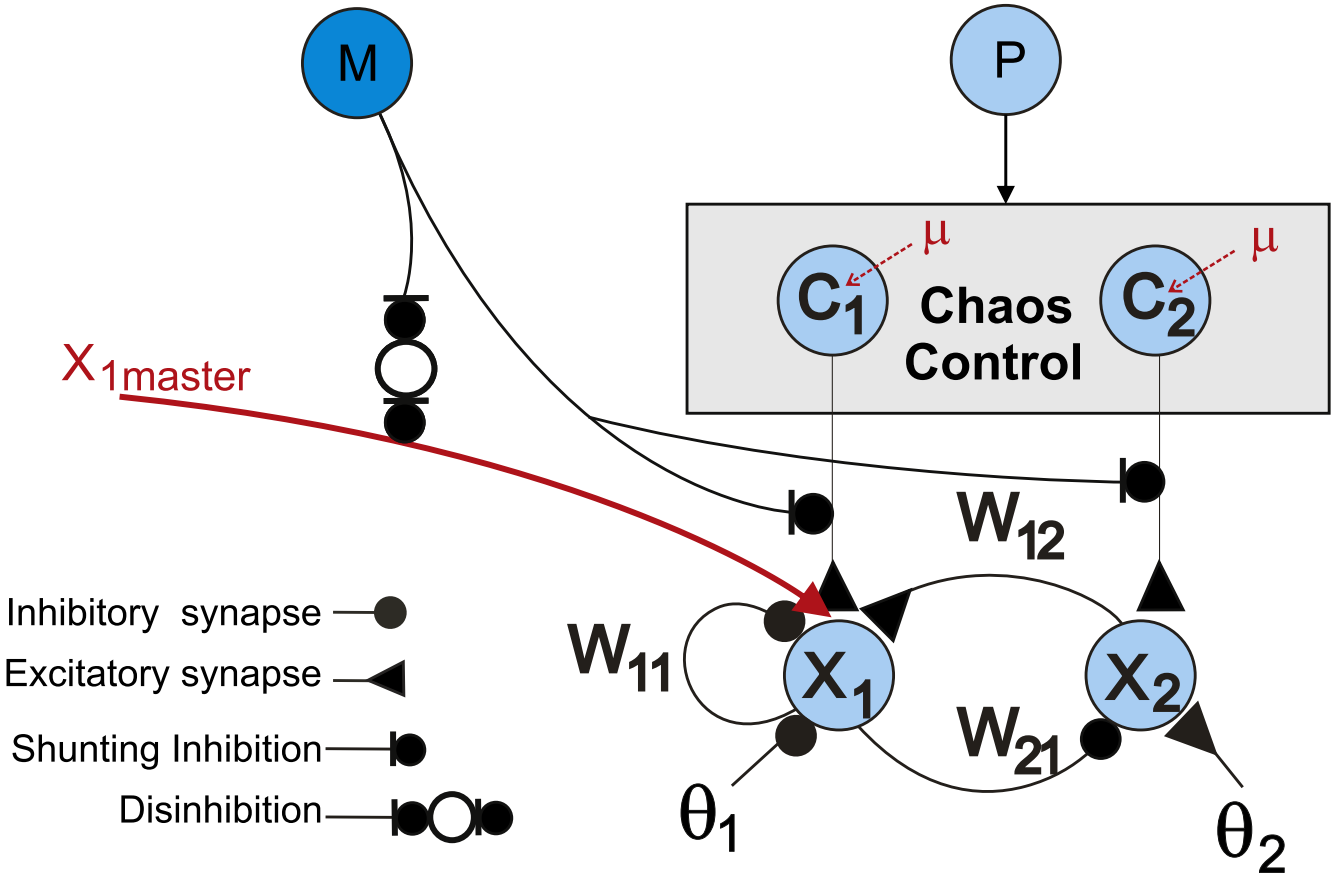
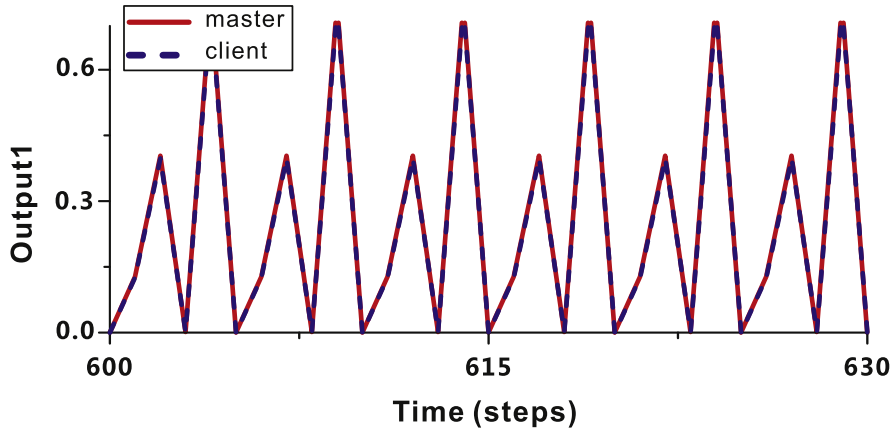
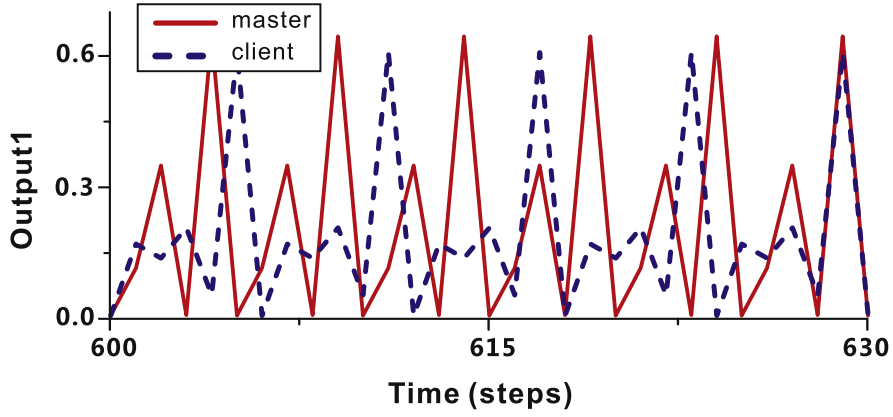


Fig. 4. The inner structure of the client CPG.

Synchronous and asynchronous CPGs



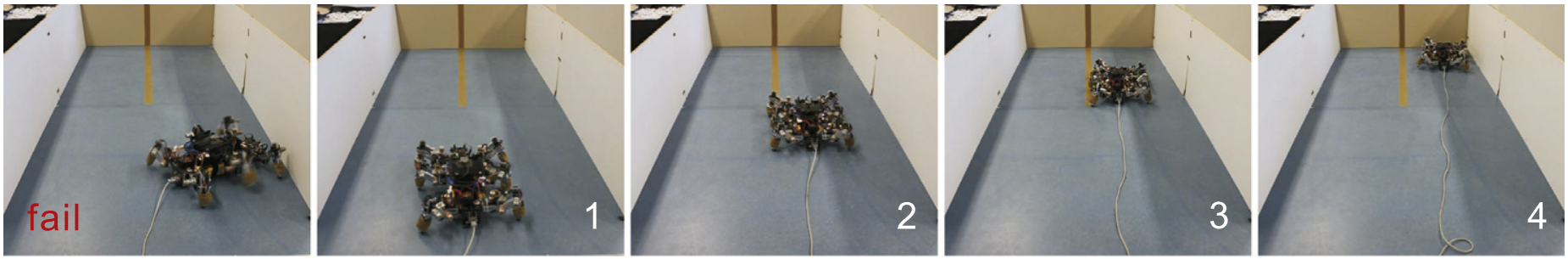
(a) Synchronous

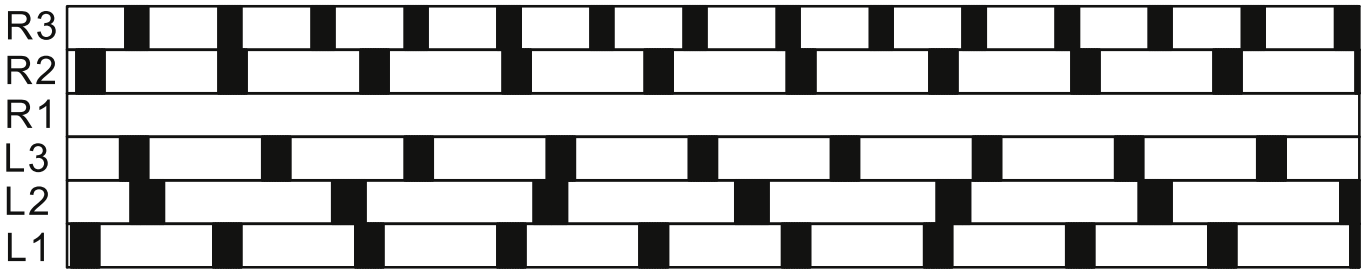


(b) Asynchronous

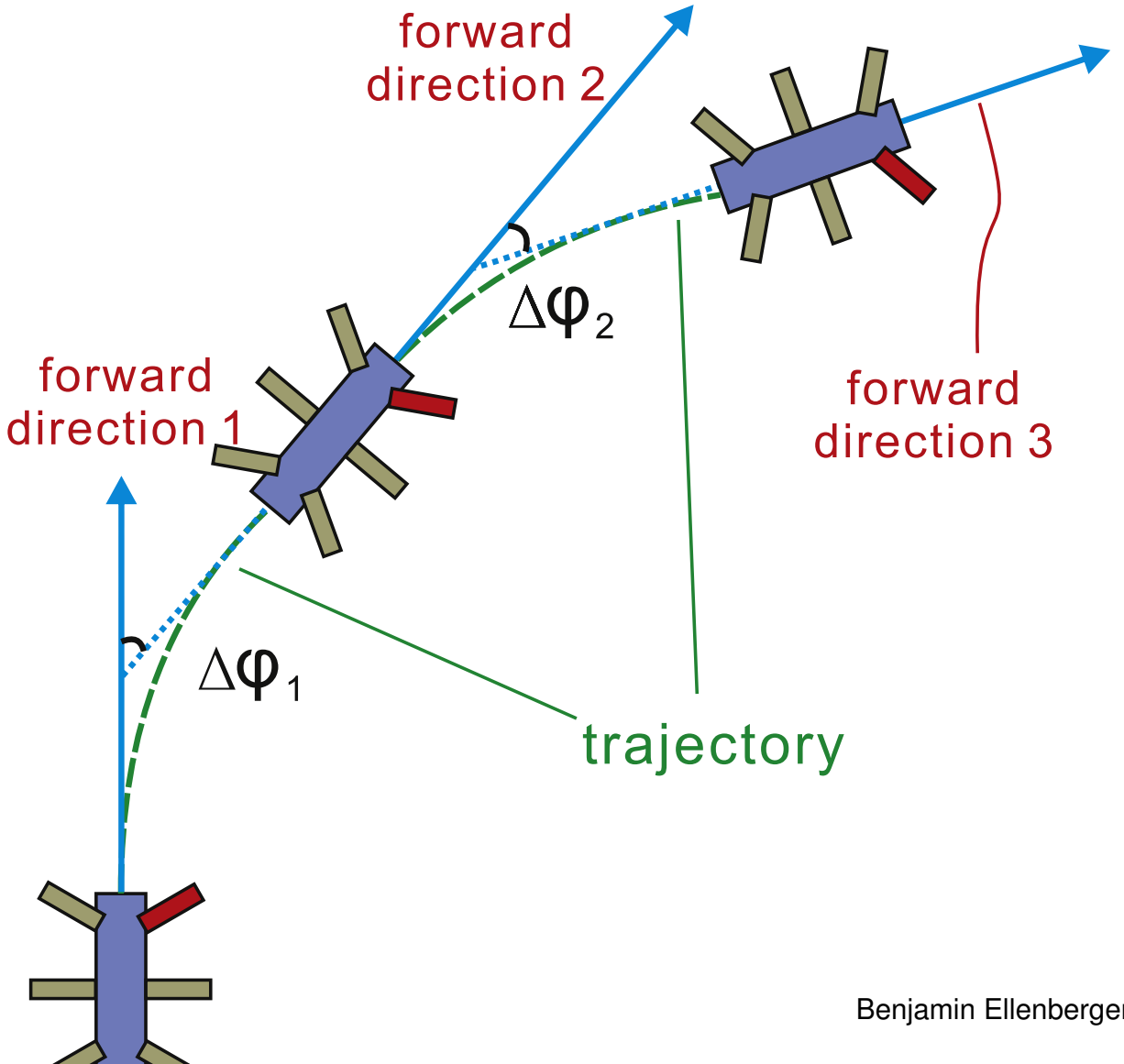
Fig. 5. The outputs of the CPG network for (a) synchrony and (b) asynchrony.

Experiment setup





Trajectory deviation measure



Simulated annealing for leg period learning

initialize $C(1) = [4; 4; 4; 4; 4; 4]; \Delta\phi = 0.0; E_1 = 0.0$
repeat:
    At repetition n
    do
        randomly pick a leg l, $l \in [R1; R2; R3; L1; L2; L3]$
        change the period of leg l to a random value, $P(l) \in [4; 5; 6; 8; 9]$
        compare this combination of leg periods, $C_{new}(n)$, to the leg periods of $C(n-1)$
    until $C_{new}(n)$ is a new combination of leg periods

    run the robot
    // calculate the evaluation function and its variation
    $E_n = \Delta\phi$
    $\Delta E = E_n - E_{n-1}$

    // choose the combination of leg periods
    if $\Delta E < 0$ ***then***
        $C(n) = C_{new}(n)$
    else
        if $X \geq e^{-\beta\Delta E}$ ***then*** // $X \in [0; 1]$, *rejection rate* β
            $C(n) = C_{new}(n)$
        else
            $C(n) = C(n-1)$
        end if
    end if
until: The evaluation function E_n is less than a required value E_{req}

Learning process

NO.	R1	R2	R3	L1	L2	L3	Deviation Angle (degree)	Decision
0	4	4	4	4	4	4	37.8604	Start
1	4	4	4	5	4	4	21.4578	Keep
2	4	4	9	5	4	4	64.1167	Return to No.1
3	4	4	4	5	9	4	14.0168	Keep
4	4	4	4	5	6	4	9.5276	Keep
5	4	5	4	5	6	4	9.6890	Keep
6	4	5	4	5	6	8	21.8642	Return to No.5
7	4	5	4	5	6	5	0.2451	End

Fig. 8. The learning process for one scenario (see text for details).

β Tuning (Changing the rejection rate)

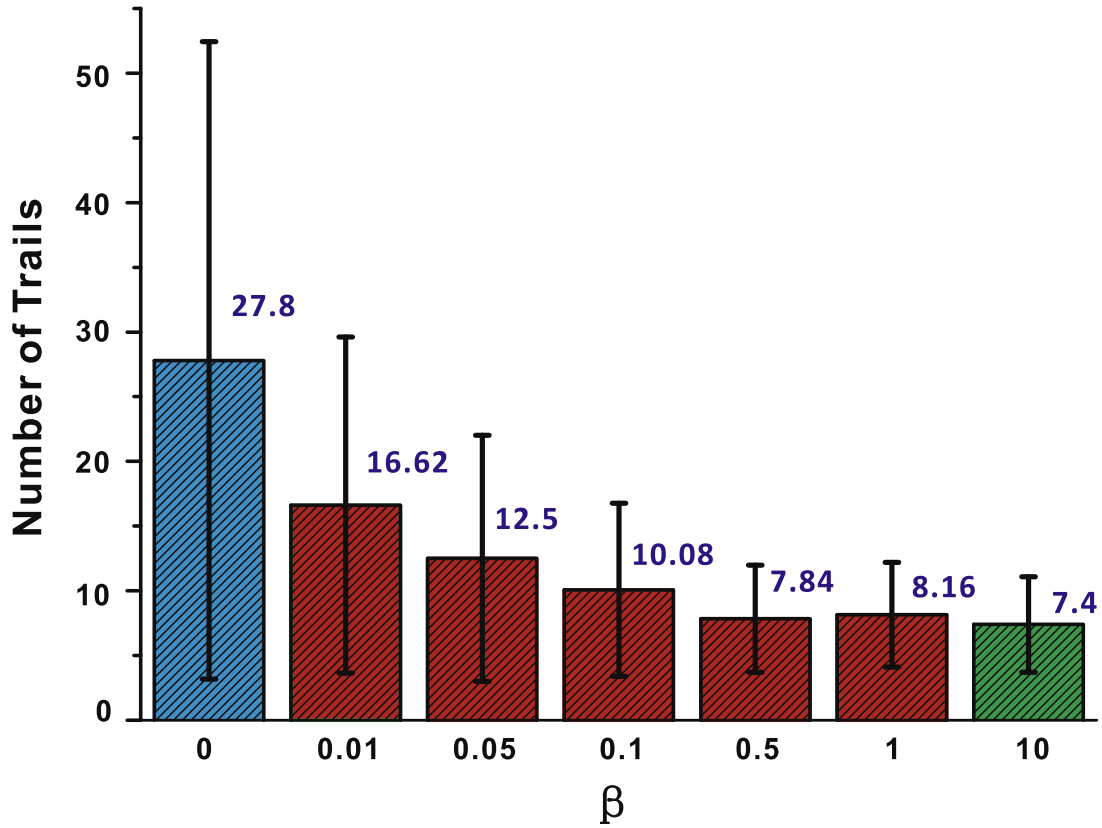


Fig. 14. The average number of trials with different β (see text for details).

Different leg disabilities

Leg status	Disabled leg(s)	Example of periods after learning	Average deviation angle (deg) + SD	Trials + SD
	R1	4, 5, 4, 5, 6, 5	3.61 ± 2.10	2.8 ± 1.99 41±29sec
	R2	4, 4, 4, 6, 4, 4	3.52 ± 2.00	1.9 ± 1.51 28±22sec
	R3	4, 4, 4, 9, 4, 4	3.45 ± 2.45	7.9 ± 9.05 117±134sec
	L1	4, 6, 4, 4, 4, 4	4.57 ± 2.30	2.4 ± 1.69 35±25sec
	L2	4, 4, 6, 4, 4, 4	4.88 ± 1.98	2.6 ± 2.11 38±31sec
	L3	4, 6, 4, 4, 4, 4	4.47 ± 1.66	4.6 ± 4.72 68±70sec
	R2, R3	4, 4, 4, 5, 8, 9	4.24 ± 2.69	8.0 ± 5.31 118±78sec
	R1, R3	4, 4, 4, 6, 4, 9	2.64 ± 2.05	6.1 ± 3.94 90±58sec
	R1, R2	4, 4, 4, 8, 9, 6	4.35 ± 1.59	7.6 ± 2.20 112±33sec
	R1, L3	4, 5, 4, 4, 5, 4	3.29 ± 2.16	10.9 ± 5.84 161±87sec
	R2, L3	8, 4, 4, 4, 8, 4	4.17 ± 2.29	10.7 ± 6.77 158±100sec
	R1, L2	4, 4, 4, 4, 4, 4	0.83 ± 0.04	0.0 ± 0.0 0±0sec
	R1,R3,L1	4, 4, 4, 4, 5, 5	4.39 ± 2.09	7.7 ± 6.51 114±96sec
	R1,R3,L3	4, 4, 4, 4, 9, 4	5.08 ± 1.29	3.6 ± 3.83 53±57sec
	R1,R3,L2	4, 4, 4, 4, 4, 9	2.01 ± 1.61	5.7 ± 5.92 84±88sec
	R1,R2,L1	4, 4, 5, 4, 8, 8	4.67 ± 2.50	4.9 ± 3.08 73±46sec
	R1,R2,L3	4, 4, 4, 5, 9, 4	3.81 ± 2.31	5.8 ± 4.04 86±60sec
	R1,R2,L2	4, 4, 4, 4, 4, 9	2.98 ± 2.21	3.5 ± 2.46 52±36sec
	R2,R3,L1	4, 4, 4, 4, 8, 5	2.49 ± 2.07	3.8 ± 2.52 57±37sec
	R2,R3,L3	4, 4, 4, 6, 5, 4	4.94 ± 2.37	3.7 ± 2.10 55±31sec
	R2,R3,LM	4, 4, 4, 4, 4, 8	5.03 ± 1.93	3.3 ± 1.55 49±23sec

Real performance video

Comparison to other approaches

	Ren & Chen	**Cully & Clune**
Performance Assessment Model	None	Gaussian Process
Additional Prior	None	Search space premodelling & prior performance values
Optimization	Simulated Annealing (GD)	Intelligent Trial & Error (Bayesian Optimization)
Trials	Few (1-10)	Few (1-10)

How can the performances be so similar?

[5] From [3] A. Cully, J. Clune et al., Nature 521 (2015)

Keypoints

- A chaotic system can be controlled into showing periodic dynamics, so as to be implemented as a CPG to accomplish the locomotion control of a bio-inspired hexapod robot
- Multiple coupled chaotic CPGs with learning can be used for legged locomotion and malfunction compensation
- A simulated annealing optimization is sufficient if the basic locomotion mechanism aids the optimization
- Chaotic ground state can improve the overall performance of a robot and make it easier to search for matching periodicities

Flaws

- The system does not use any additional sensors to detect malfunction.
- The PSN/VRN networks and the delay lines make it a beautiful walking machine, but unfortunately make it unable to do anything else.
- Broken legs are generally not disabled totally, but misbehave in some way (Reduced torque output, incorrect position etc.). This case was not considered.

Discussion!

- Any questions?

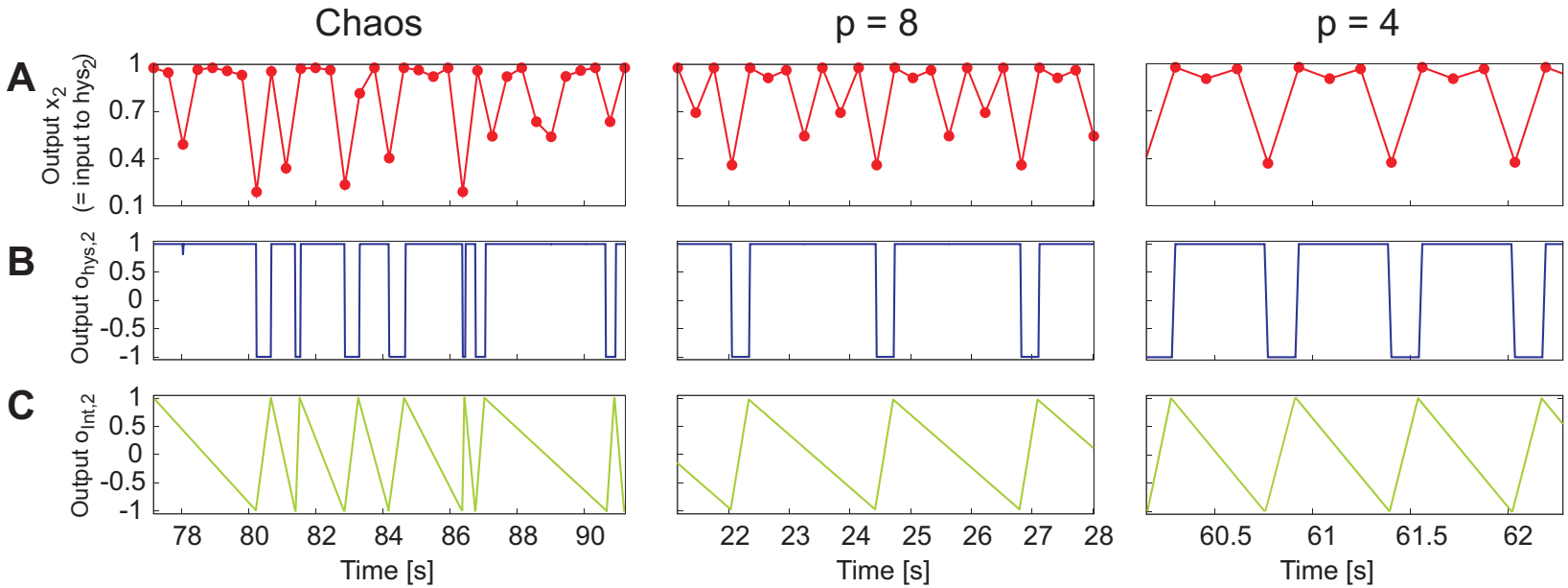
References

[1] G. Ren, W. Chen. et al., "Multiple chaotic central pattern generators with learning for legged locomotion and malfunction compensation", Information Sciences 294 (2015)

[2] S.Steingrube, M. Timme et al., "Self-organized adaptation of a simple neural circuit enables complex robot behaviour", Nature Physics 6 (2010)

[3] A. Cully, J. Clune et al. "Robots that can adapt like animals", Nature 521 (2015)

Postprocessing motor neuron signals



Supplementary Figure 3: **Postprocessing signals for different periods.** **(A)** Output signals of the time window function unit Δt. **(B)** Output signals of the hysteresis unit $hys2$. **(C)** Output signals of the signal integrator unit $Int2$.

[6] From [2] S.Steingrube, M. Timme et al., Nature Physics 6 (2010)