Models of Biological Sensory-Motor Systems

Modeling and Simulating the Salamander's Locomotion and Visual System

Project Report

Rafael Arco Arredondo
rafael.arcoarredondo@epfl.ch

Jean-Philippe Pellet
jean-philippe.pellet@epfl.ch

February 19, 2006

Table of Contents

1 Introduction **3**
1.1 Simulation Framework . . . 3

2 Locomotion with a Sine-Based Controller **3**
2.1 Description . . . 3
2.1.1 Swimming Gait . . . 3
2.1.2 Walking Gait . . . 4
2.1.3 Gait Transitions . . . 5
2.2 Analysis . . . 6
2.2.1 Walking Gait . . . 6
2.2.2 Swimming Gait . . . 9

3 Locomotion with a Central Pattern Generator-Based Controller **9**
3.1 The Simple Controlled-Energy Oscillator . . . 11
3.2 Coupled Controlled-Energy Oscillators . . . 11
3.3 Implementation and Numerical Evaluation . . . 12
3.4 Setting the Parameters: Genetic Algorithm . . . 13
3.4.1 Results of the Genetic Algorithm . . . 14
3.5 Setting the Parameters: Systematic Search by Hand . . . 15
3.5.1 Results of the Systematic Search . . . 16

4 Visual System **17**
4.1 Description of the Model . . . 18
4.1.1 Retina . . . 19
4.1.2 Optic Tectum . . . 19
4.1.3 Brain Stem . . . 20
4.2 Tests: Tracking the Fly . . . 22

5 Conclusion **22**

References **23**

1 Introduction

Understanding Nature helps us human beings understand the world in which we live, and ultimately helps to understand ourselves and the way we function. Studying simple examples involving less complicated modeling helps to decompose a complex system into smaller parts whose individual complexity is reasonably low.

In this project, we used a bio-oriented approach to model and simulate some parts of a salamander's sensory-motor system, namely its locomotion control, involving two different gaits (swimming and walking), and its visual system, whose goal will be in our case to influence the locomotion control system in order to try and follow a randomly-moving fly.

We first discuss a naive approach to locomotion control via a simple sine-based controller. We show its weaknesses and motivate why a more elaborate model based on central pattern generators works better. Finally, we go over to vision, detail a simple model involving retina, tectum and brain stem, and show how we coupled the output of the visual system to the locomotion control system. We explain a few implementation details, as well as why some particular techniques were succeeding or failing at solving the particular problems we encountered.

1.1 Simulation Framework

We worked with Webots [11, 13], mobile robot simulation software from Cyberbotics, on an artificial salamander robot [9] developed by the Biologically-Inspired Robotics Group group at EPFL, lead by Prof. Auke Jan Ispeert. The salamander is made up of 9 body parts and 4 rotating legs as shown in Figure 1 and is about one meter long.

2 Locomotion with a Sine-Based Controller

2.1 Description

The salamander must be able to walk and swim in a way close to its natural gaits (movies available from [1]). The sine-based controller basically switches between these two gaits, each of which is described with a sine-based equation.

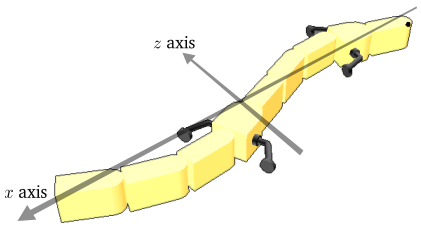

FIG. 1: The salamander robot and its local axes

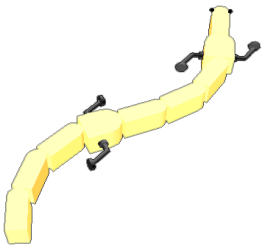

FIG. 2: The salamander while walking

2.1.1 Swimming Gait

Continuous Description We based our swimming gait controller on a sine wave propagating along the salamander's body. It was shown [6] that similar organisms such as the lamprey use this technique of the propagating wave to achieve locomotion in water; additional work [7, 10], suggest that this can be applied to salamanders, too. A first try with a simple sine function could describe the form of the salamander body like this:

$$z(x,t) = A_s \sin\left(\frac{2\pi}{L}x + 2\pi f_s t\right) \tag{1}$$

We assume a 2D coordinate system (x, z) relative to the salamander as depicted in Figure 1, where the x axis is pointing from the head to the tail, parallel to the current direction, and the z axis is perpendicular to x in the plane we consider. We thus neglect any y coordinate for now, we don't model height. L is the length of the body of the salamander projected on the x axis.

Parameters are A, the body amplitude, and f, the oscillation frequency. Adding the term $\frac{2\pi}{L}$ ensures that the body always forms a complete period (as far as the amplitude is not too high). We could consider non-constant amplitude $A = A(\frac{2\pi}{L}x)$, e.g. in order for the tail to have bigger amplitude than the head, but for now we'll use a constant amplitude.

Discretization The salamander robot is actually made of 9 parts, which we'll call p_0 to p_8: one for the head, one supporting the two front legs, three representing the body between the two leg pairs, one supporting the two rear legs and finally three more to represent the tail. There are therefore eight junctions between them, which we'll call j_1 to j_8 where j_i is the junction between parts p_{i-1} and p_i. Of these eight junctions, six are mobile and can be controlled with servos. We'll call (the current value of) these servos s_i^b where $i \in S^b = \{2, 3, 4, 6, 7, 8\}$. The time t is given as an integer representing milliseconds. We can now discretize the model from Equation 1 by changing the dependance on x into a phase shift ϕ_i defined for each $i \in S^B$:

$$s_i^b(t) = A_s \sin\left(2\pi f_s \frac{t}{1000} + \phi_i\right) \tag{2}$$

$$\phi_i = \frac{2\pi}{7}(7 - i) \tag{3}$$

The total phase shift between the first and the last servo should be 2π in order to see a full oscillations at all times in the shape of the body. The index i runs up to 7, so we divide a full period by 7 and multiply by i—or by $(7 - i)$ in our case to get a travelling wave in the correct direction.

While swimming, the legs of the salamander do not move and are oriented backwards.

2.1.2 Walking Gait

Body Similarly to what we did with the swimming gait, we can define a continuous version of what the salamander body should look like when walking:

$$z(x, t) = A_w \sin\left(\frac{2\pi}{L}x\right) \sin\left(2\pi f_w^b t\right) \tag{4}$$

This time the function that controls the movement of the body is slightly different. Although it is still a sine-based function, now it does not oscillate in the same way as the the swimming gait, but it makes the salamander's body move with the trunk in antiphase respect to the tail (which is approximately the gait a real salamander performs when walking).

The discretized version becomes:

$$s_i^b(t) = A_i \sin\left(2\pi f_w^b \frac{t}{1000}\right) \tag{5}$$

$$A_i = A_w \sin\left(\frac{2\pi}{7}i\right) \tag{6}$$

Legs The movement of the legs is really simple. They just turn at a constant speed, having a linear dependance with respect to the time. This is not possible of course for a real salamander, but it is the only way in the robot to not touch the ground in a backwards movement, as it would happen for instance with a sine-based gait, with the consequent loss of speed. These are the equations that describe the movement of the legs:

$$s_i^l(t) = \begin{cases} 2\pi f_w^l \cdot t/1000 + \varphi & \text{if } i \in \{1, 4\} \\ 2\pi f_w^l \cdot t/1000 + \varphi + \pi & \text{if } i \in \{2, 3\} \end{cases} \tag{7}$$

The term φ is used to produce a difference of phase between the movement of the legs and the movement of the body. Note that the front-left and front-right legs are in phase with the rear-right and the rear-left ones, respectively, while they are in antiphase with the other ones (a term of π radians is added to the phase). Salamanders always walk in this way, because the speed is higher and the equilibrium is better maintained.

2.1.3 Gait Transitions

The salamander now checks its current vertical position to know which gait (swimming or walking) it must adopt. Tests show that a simple gait threshold θ_{gait} is not a very good solution, for either the salamander starts swimming too early and thus cannot enter the water well, or it cannot properly go out of the water because the legs start moving when they are not touching the ground yet.

Therefore, a state machine modeling the gait was implemented and two thresholds were set, $\theta_{w\to s}$ determining when the salamander must switch its state from WALKING to SWIMMING, and $\theta_{s\to w}$ determining when the salamander must stop swimming to begin walking. It holds:

$$\theta_{w\to s} < \text{water level} < \theta_{s\to w}$$

Namely, there is an interval $(\theta_{w\to s}, \theta_{s\to w})$ in which both gaits are possible, depending on the current state.

Of course, the underlying assumption is that the salamander will be in the water if and only if its vertical coordinate is below the water level parameter (this must hold in the whole world) and on the ground otherwise. In practice, the salamander's vertical coordinate is approximated by retrieving the vertical coordinate of one of the middle body servos.

Amplitude Change There are two different maximal amplitudes for walking and swimming: A_w and A_s, respectively. For the moment, we instantly switch from one to the other. This is not very good with the current sine-based controller as it introduces jumps in the target values of the servos. This is one of the shortcomings of this controller and will also be one of the reasons to look for another, more tolerant controller in section 3. Meanwhile, we can try to make our sine-based controller react more smoothly to parameter changes.

We could imagine transitions happening over a transition time t_{trans}. The following equation would for instance model a linear transition (A_1 is the amplitude the transition starts with and A_2 is the target amplitude; we assume the transition to happen at $t = 0$):

$$A_{linear}(t) = \begin{cases} A_1 + t \cdot (A_2 - A_1)/t_{trans} & \text{if } t < t_{trans} \\ A_2 & \text{if } t \geq t_{trans} \end{cases} \tag{8}$$

Or an even smoother, cosine-based transition:

$$A_{cosine}(t) = \begin{cases} A_1 + \frac{1}{2}(A_2 - A_1)\left(1 - \cos\left(\pi t / t_{trans}\right)\right) & \text{if } t < t_{trans} \\ A_2 & \text{if } t \geq t_{trans} \end{cases} \tag{9}$$

Note that if the salamander were to switch e.g. from SWIMMING to WALKING and then to SWIMMING back again in a time $t_s < t_{trans}$, then A_1 would be substituted in Equations 8 and 9 with the currently reached transition amplitude $A(t_s)$.

Figure 3 shows the adaptations of the amplitude. While Equation 8 does avoid sudden jumps in the servos' target positions, it is hardly noticeable in the simulation. The changes brought by using Equation 9 rather than Equation 8 cannot be observed.

Frequency Change Ideally, the frequency transition should also be a smooth one, in order to avoid some awkward movement as the salamander gets into or out of the water. However, it cannot be changed adaptively in a way similar to the one proposed for the amplitude transitions, as changing

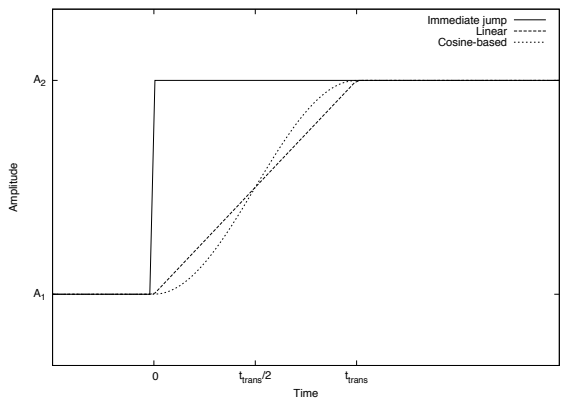

FIG. 3: Amplitude transitions

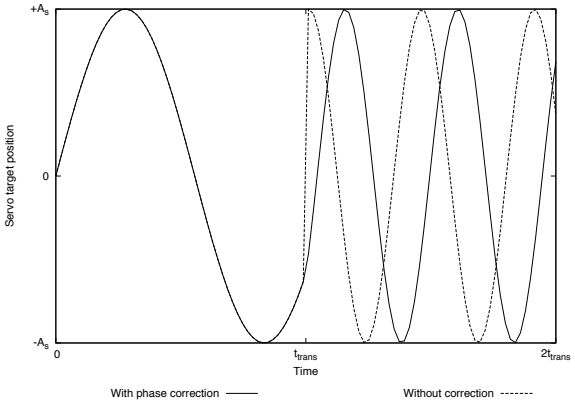

FIG. 4: Frequency transitions

the frequency will usually mean arriving at a different moment of the sine cycle and will cause jumps in the servos' target positions.

A possible workaround would be to add a new additional phase shift to Equation 3, which would become:

$$\phi_i = \frac{2\pi}{7}(7 - i) + \varphi \tag{10}$$

The term φ would be set to 0 initially and would be changed each time the frequency would be changed too. Its purpose is to ensure that Equation 2 remains continuous (although not its derivative) even if f_s changes, and would be computing like so (with a frequency change happening at $t = t_{trans}$):

$$\varphi_{new} = 2\pi(f_{s,old} - f_{s,new})\frac{t_{trans}}{1000} \tag{11}$$

Figure 4 shows the uncorrected and phase-corrected frequency changes.

Despite this trick, we can say that the sine-based controller is not really appropriate for easy frequency (and thus speed) tuning because of the jumps induced in the equations describing the servos' target positions. We can try to smooth this behaviour by introducing correcting parameters like it is described here, or we can use a more complex controller, for instance based on a set of differential equations. In this case, sudden changes in the model's parameter would introduce jumps only in the derivative of the function describing the target values of the servos and not in the function itself. See section 3 for further details.

2.2 Analysis

2.2.1 Walking Gait

In order to optimise the sine-based controller, a study of the walking gait must be performed. We varied the parameters that describe the movement of the salamander and analysed how the changes on them affect the linear velocity and the amplitude of the body.

Linear velocity vs. frequency The linear velocity is of course affected by the frequency of the wave equation of the salamander. Normally, the speed increases with the a higher frequency, although this is not always true, as can be seen in Figure 5.

To generate this plot, we varied the frequency (both f_w^l and f_w^b, and setting the restriction $f_w^l = f_w^b$[1]) from 0 rad/s to the maximum speed the servos can reach, which is 20 rad/s. We fixed the body amplitude

[1]This is generally true for all kinds of locomotion: there is only one main frequency, and possibly multiples of this frequency, that is used for the legs and the body.

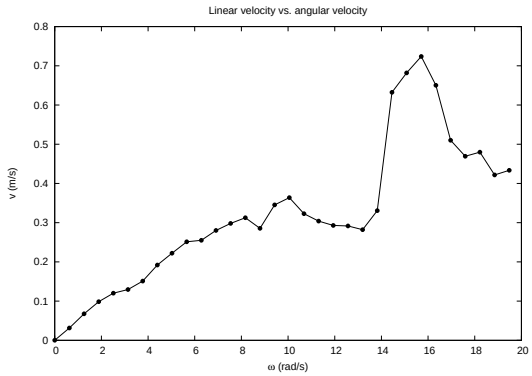

FIG. 5: Linear velocity vs. angular velocity

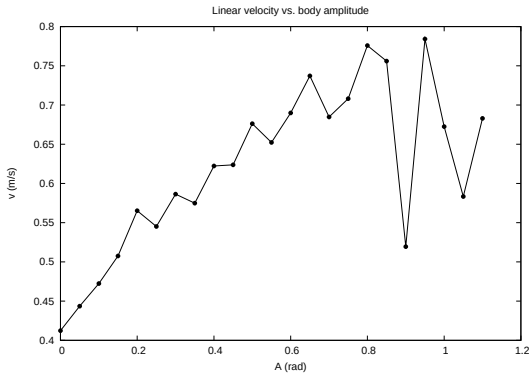

FIG. 6: Linear velocity vs. body amplitude

to 0.75 rad and the legs in phase with the body, putting the front-left leg on the ground when the trunk turned to left and vice versa.

We can see in the plot that the speed increases smoothly until approximately 10 rad/s, reaching 36 cm/s. Then it goes down a little until 13.82 rad/s, for which the speed is 33.06 cm/s, and later there is a big jump that makes the linear velocity go up getting a peak of 72.39 cm/s for an angular velocity of 15.71 rad/s.

This increase of more than twice the speed in less than 2 rad/s was rather funny for us, so we took a closer look to the trajectory of our salamander. What happened was that the salamander, with low frequencies, moves really slowly because there is a great amount of time where no foot is pushing the ground, which is the main factor to go ahead. With higher frequencies (higher than 14 rad/s), this time is drastically reduced.

Finally, beyond 16 rad/s the speed decreases very quickly due to a bad coordination, because the servos (either the leg servos or the body servos) do not have time to reach the desired position, so the movement is "interrupted" with some abrupt jumps (sometimes, the direction of the servos changes).

Linear velocity vs. body amplitude Amplitude also plays an important role in the definition of the movement of the salamander. In order to study the behaviour against the amplitude, we varied it until the maximum position of the servos of the body, 1.13 rad. We kept a constant angular velocity of 15.71 rad/s and put the legs in phase with the body again. Figure 6 summarises this exploration.

A clearly increasing trend in the linear velocity can be seen as the amplitude is bigger. This is normal because the oscillation of the body helps to push the salamander ahead. Velocity gets higher and higher until the amplitude reaches 0.95 radians. At this point, the speed is 78.41 cm/s. Then the speed decreases because the stability of the gait gets worse.

Linear velocity vs. body amplitude and frequency In order to complete our analysis about the frequency and the amplitude, we wanted to study the influence of the combination of both parameters over the linear velocity of the salamander, to expand the search space because maybe our previous values were local optima. We varied the parameters along the allowed values (from 0 rad to 1.13 rad for the amplitude and from 0 rad/s to 20 rad/s for the angular velocity) and got the plot showed in Figure 7.

The plot shows that frequency influences much more the speed of the salamander than amplitude. In fact, amplitude only seems to be important when frequency has a value which is close to the ones that provide the highest linear velocity (around 15 rad/s). For these values, the speed of the salamander increases as amplitude does the same thing. We found this time a maximum speed of 83.51 cm/s for an amplitude of 1 rad and an angular velocity of 15.71 rad/s. This value is different from the previous one that we found for the same angular velocity, but this is reasonable because there are a lot of factors that

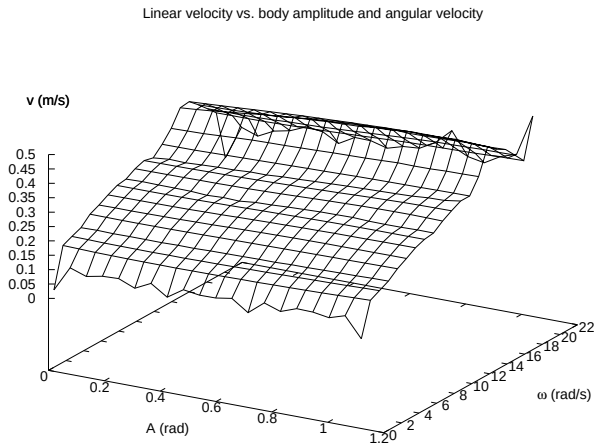

FIG. 7: Linear velocity vs. body amplitud and angular velocity for the walking gait

can influence on it: if the salamander turns or not,[2] the possible integration errors during the simulation, the previous values the amplitude and the frequency had...

Linear velocity vs. body-legs phase shift Another important parameter is the phase shift between the movement of the legs and the movement of the body. The salamander does not walk with the same speed if, for instance, it puts the front-left leg on the ground when the trunk is oriented to the left too or if it puts the front-left leg on the ground when the trunk is oriented to the right, or looking ahead... In Figure 8 we can see a summary of the measurements concerning this analysis.[3]

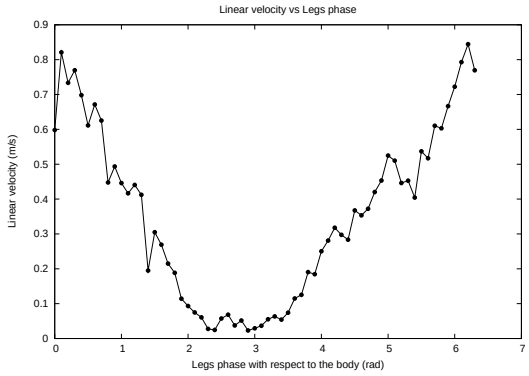

FIG. 8: Linear velocity vs. body-legs phase shift

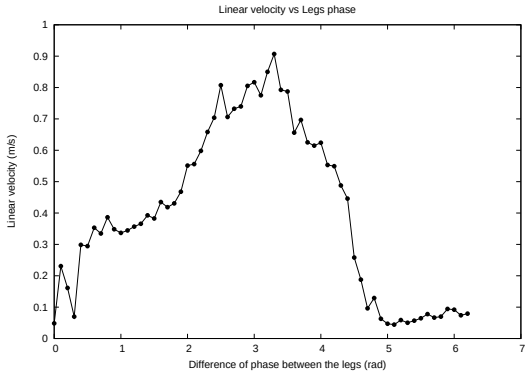

FIG. 9: Linear velocity vs. inter-leg phase shift

The plot illustrates that the best strategy in order to achieve the highest speed is to put the front-left (front-right) left on the ground when the trunk is oriented to the left (right). The speed really decreases when the coordination is the opposite (the speed is less than the 10% of the maximum value). The maximum speed we found was 84.42 cm/s, corresponding to a body-legs phase shift of 6.20 rad.

[2] In this case, the speed is lower because we took into account only the position vector instead of the whole trajectory. We think this approach is good because we are rewarding the gaits that can follow a straight line (we are not interested in turning so far).

[3] In the plot, a phase shift of 0 radians means that the front-left leg is in contact with the ground when the trunk reaches the maximum curvature oriented to the left (with the head of the salamander looking to the left as well), while a phase shift of $\pi/2$ means the front-left leg is in contact with the ground when the trunk reaches the maximum curvature oriented to the right.

Linear velocity vs. inter-leg phase shift In addition to have a good coordination between the body and the legs, it is important to have the legs well coordinated with respect to the other ones. We did all the previous experiments with the legs moving in the following way: the front-left leg is in phase with the rear-right one, as well as the front-right leg with the rear-left leg, and every leg is in dis-phase with the other two legs with which it is not in phase.

However, and although real salamanders walk with the coordination mentioned above, we made the analysis of this parameter to find the optimum value in case it was different. Figure 9 shows the results about the inter-leg phase shift.

Indeed, we can see that the highest speed is not reached with an inter-leg phase shift of exactly π radians, but a little bit more, 3.30 rad.

Desired/real amplitude vs. frequency Our analysis of the walking gait concludes taking a closer look to the differences between the desired amplitude and the one that is finally reached in the simulation, depending on the frequency of the travelling wave. When the simulations are run it can be easily seen that there is a difference between the amplitude of the wave equation that we fixed and the actual maximum amplitude the servos reach. The higher the frequency, the higher this difference is.

What happens is that the servo need some time to go from one position to the following one in the next simulation step, which is given by the following equation of a PID (Proportional Integral and Derivative) controller:

$$T(t) = K_p \cdot e(t) + K_i \cdot \int e(t)\mathrm{d}t + K_d \cdot \frac{\mathrm{d}}{\mathrm{d}t} \cdot e(t) \tag{12}$$

Where $T(t)$ is the torque applied by the servo at time t, $e(t) = \theta_{comm}(t) - \theta_{real}(t)$, that is, the difference between the commanded position and the real one, and K_p, K_i and K_d are constants representing the gain of the signal.

Therefore if the frequency is too high, the servos are not able to follow the rhythm of the changes and a delay that accumulates in each step appears. This way, the servos can go in one way while the desired position is doing *the way back* in the opposite direction. And finally the servos change their direction to reach the new desired position. In conclusion, if the frequency is too high, the servos never reach the maximum position that the desired amplitude gives, so the real amplitude of the travelling is smaller.

Figure 10 provides a graphical explanation of this fact and shows the differences between the amplitudes that several frequencies generate.

2.2.2 Swimming Gait

A similar analysis of the swimming gait was performed. During swimming, the two main relevant parameters are A_s and f_s from Equation 2. The whole parameter space exploration can be summed up with Figure 11.

The body amplitude was varied from 0.2 rad to the maximum value of 1.13 rad. During swimming, the achieved velocity monotonically increases with the body amplitude; we can then say that the maximal amplitude is optimal. We must not think, however, that the maximum amplitude is actually reached by the servos: their real value has always a little delay which increases with the frequency and ultimately causes the actual amplitude to be smaller, as shown in the previous graphs.

On the other hand, a higher frequency does not always mean a higher velocity (because of the previously described effects). We can easily see on Figure 11 that the optimal velocity is not achieved with the highest frequency; our optimal frequency is about 0.03 rad/s with a velocity of about 0.93 m/s.

3 Locomotion with a Central Pattern Generator-Based Controller

In this section, we abandon the sine-based controller because of its shortcomings and switch to a more bio-inspired locomotion controller based on a central pattern generator (CPG) principle. Biological evidence suggests that in nature, locomotion is governed by rhythms due to a central mechanism [5] rather

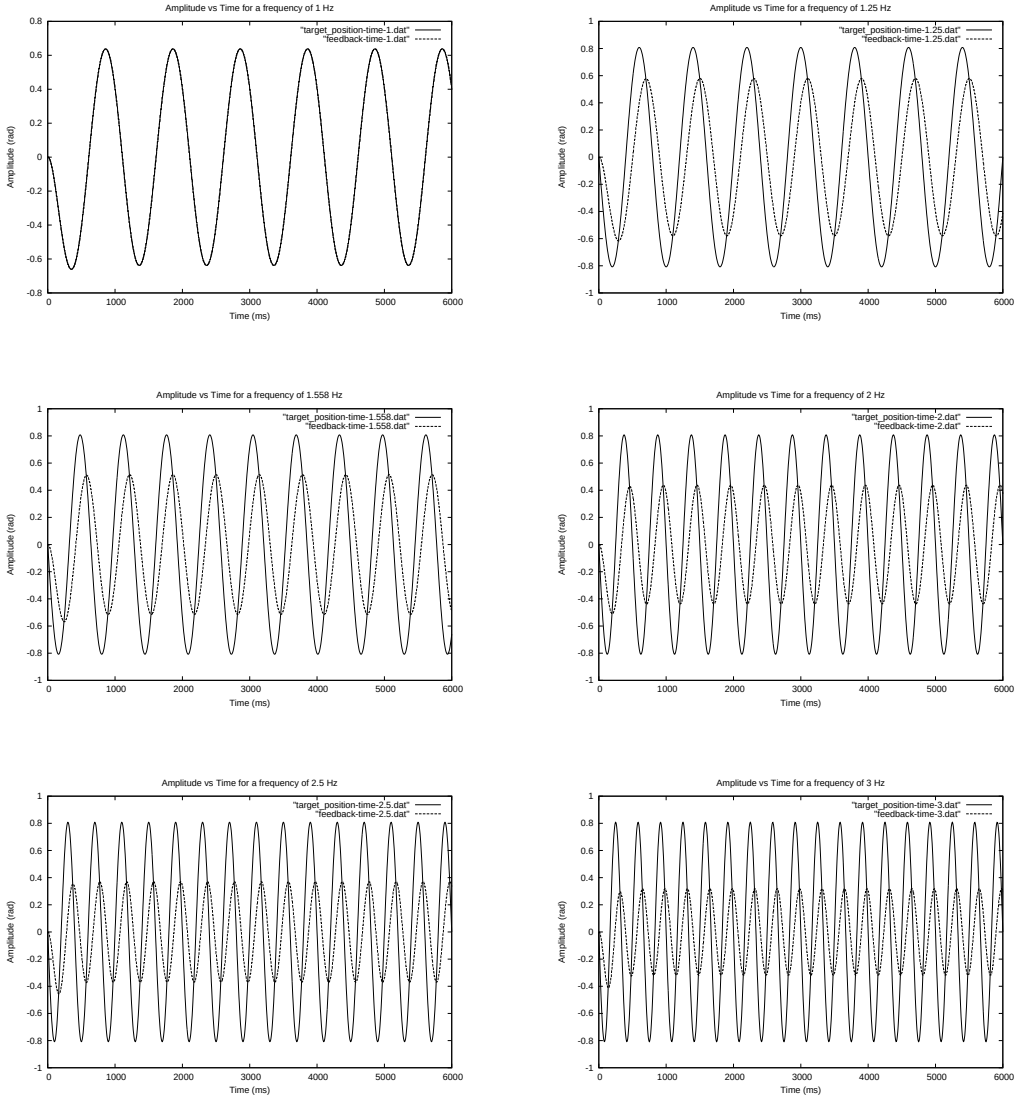

FIG. 10: Desired and real amplitude vs. frequency

than to a chain of reflexes involving the peripheral system [12]. As shown by Grillner [6], these rhythms do not need sensory feedback to be generated; they are, however, strongly modulated by it.

Non-linear oscillators described by a set of differential equations are particularly good candidates for smooth gait transition, frequency and/or amplitude adaptation, and prevent state variables from jumping from one value to the other, as this was the case with the sine-based controller, e.g. without the explicit phase correction described by Equation 11.

Adequate oscillators are for instance the Hopf oscillator [2] or the controlled-energy (CE) oscillator, introduced for locomotion by Ijspeert et al. in [9]. Other methods include networks of leaky-integrator neurons (model described e.g. in [3]; network type and connections for locomotion described in [8]). In this project, we decided to use a chain of coupled CE oscillators.

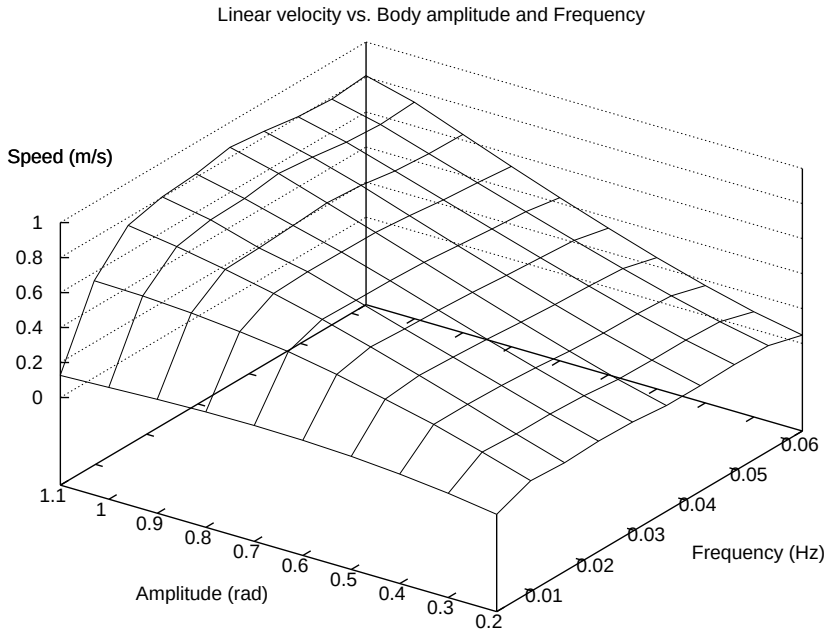

FIG. 11: Linear velocity vs. body amplitude and frequency for the swimming gait

3.1 The Simple Controlled-Energy Oscillator

The state of a simple CE oscillator as described by [9] is fully characterized by the two states variables x and v which obey the following differential equations:

$$\tau \frac{dv}{dt} = -\alpha \frac{x^2 + v^2 - E}{E} v - x \tag{13}$$

$$\tau \frac{dx}{dt} = v \tag{14}$$

E, τ and α are parameters. This oscillator converges to the limit cycle for x:

$$\tilde{x}(t) = \sqrt{E} \sin\left(t/\tau + \phi\right) \tag{15}$$

where the phase ϕ depends on initial conditions. τ thus regulates the oscillation frequency and E its amplitude. Finally, α tunes how fast the limit cycle is reached, and subsequently how reactive the oscillator is with respect to changes affecting other parameters.

3.2 Coupled Controlled-Energy Oscillators

In order to couple several CE oscillators, we introduce two new parameters per coupling, a_{ij} and b_{ij}, which model a coupling from oscillator i to oscillator j. The state variables for oscillator i are now called x_i and v_i and obey these modified equations:

$$\tau_i \frac{dv_i}{dt} = -\alpha_i \frac{x_i^2 + v_i^2 - E_i}{E_i} v_i - x_i + \sum_j \left(a_{ij} x_j + b_{ij} v_j\right) \tag{16}$$

$$\tau_i \frac{dx_i}{dt} = v_i \tag{17}$$

In the most general case, E_i, τ_i and α_i can be all different, but in our case, we keep the same value for $\tau_i = \tau$ and $\alpha_i = \alpha$. The oscillators' amplitude however can still be set individually, so it does not hold that $E_i = E_j \; \forall i, j$.

We want to map these oscillators to the salamander's body servos. In nature, each muscle controlling the shape of the body on the right side of the body has its antagonist, symmetrical muscle on the left side. When a muscle contracts, its antagonist must relax: that is how the shape of the body can vary.

To model these pairs of antagonist muscles, we build a chain of pairs of oscillators. Each oscillator is coupled to its antagonist oscillator, to the previous oscillator in the chain and to the next one. Because of left/right symmetry, we set the couplings from the right oscillator to the left one to be equal; furthermore, we define all oscillator pairs to be coupled in the same way with the previous and next pair.

Parameters We thus reduce the number of coupling parameters to 6, which we rename a_{side}, b_{side}, a_{prev}, b_{prev}, a_{next} and b_{next}. This is illustrated in Figure 12.

To these parameters we also add E_i (one per oscillator), α and τ. We have 6 servos, we thus need 12 oscillators. We add two pairs to model the segments which are fixed and whose angle cannot be controlled by servos to be closer to our salamander model: we end up with 16 oscillators. The total number of parameters to set adds up to $6 + 16 + 2 = 24$.

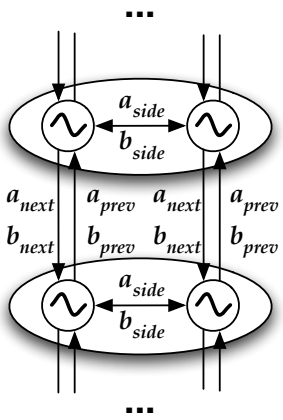

FIG. 12: Oscillator pair couplings

3.3 Implementation and Numerical Evaluation

In the C implementation of the oscillators, we use the two-dimensional 4th order Runge-Kutta numerical integration method. If we rewrite Equations 16 and 17 as follows (omitting the indices i for readability):

$$\frac{dv}{dt} = f(v, x) \qquad \frac{dx}{dt} = g(v) \tag{18}$$

then, knowing x and v at a given time t, we can approximate them at time $t + \Delta t$:

$$v(t + \Delta t) = v(t) + \frac{\Delta t}{6}(k_1 + 2k_2 + 2k_3 + k_4) \tag{19}$$

$$x(t + \Delta t) = x(t) + \frac{\Delta t}{6}(j_1 + 2j_2 + 2j_3 + j_4) \tag{20}$$

$$k_1 = f(v(t),\ x(t)) \qquad j_1 = g(v(t)) \tag{21}$$

$$k_2 = f(v(t) + \frac{\Delta t}{2}k_1,\ x(t) + \frac{\Delta t}{2}j_1) \qquad j_2 = g(v(t) + \frac{\Delta t}{2}k_1) \tag{22}$$

$$k_3 = f(v(t) + \frac{\Delta t}{2}k_2,\ x(t) + \frac{\Delta t}{2}j_2) \qquad j_3 = g(v(t) + \frac{\Delta t}{2}k_2) \tag{23}$$

$$k_4 = f(v(t) + \Delta t \cdot k_3,\ x(t) + \Delta t \cdot j_3) \qquad j_4 = g(v(t) + \Delta t \cdot k_3) \tag{24}$$

In our implementation, we have the integration step $\Delta t = 5$ ms.

3.4 Setting the Parameters: Genetic Algorithm

As described in subsection 3.2, we need to set 24 parameters in order to have a complete description of our set of coupled oscillators and therefore get a traveling wave for the salamander.[4] The parameters which best suit in order to get this objective are in principle unknown. Therefore they must be calculated somehow.

We only know that we would like to get a traveling wave as similar as possible to the gait of the real salamander, that is:

- *S*-like shape when walking and *fish*-like shape when swimming, as commented in subsection 2.1.
- Coordination between the movement of the legs and the movement of the body and coordination among the four legs, in order to maximize the speed of the salamander.
- Transitions between walking and swimming as smooth as possible.

What we have thus is nothing but an optimisation problem. There are several ways to solve it. One is to mathematically solve the equations that describe these restrictions in terms of the differential equations of subsection 3.1. Another one is to perform a manual, systematic search over the space of values that the parameters can have and look for the combination that better fits these requirements. Finally, it is also possible to automatize this search by using a metaheuristic procedure such as simulated annealing, evolutionary algorithms, scatter search, etc. We decided first to follow this last approach, in particular to use a genetic algorithm (GA), because it was the most biologically-inspired way to face the problem.

Genetic algorithms have been extensively described in many documents such as [4]. It is not the objective of this report to explain the way they work so we comment the particularities of our GA directly.

Genome The genome of our salamander individuals is composed of 21 of the 24 parameters we have to set to fully define the CPG. This values are stored as an array, in the way described in Figure 13.

a_{next}	b_{next}	a_{side}	b_{side}	E_0	E_1	$\cdots$	E_{15}	τ

FIG. 13: Genome used for the genetic algorithm

The other 3 parameters are set to fixed values: a_{prev} and b_{prev} to 0, that is, there are no couplings from one oscillator to its previous oscillator in the chain (the couplings are only unidirectional from head to tail), and α to 1.0, which seems to be the value that provides the best results always (with any combination of the other parameters).

Fitness function The fitness function we use to evaluate our population of salamanders is simply the distance they cover in 30 seconds. That is:[5]

[4] In addition to these 24 parameters, some other parameters to control the servos of the legs have to be added for walking.

[5] To compute the distance we take into account only the x and z coordinates. y is the vertical coordinate and it is not useful for this purpose.

$$fitness = \sqrt{(x_{t=30} - x_{t=0})^2 + (z_{t=30} - z_{t=0})^2} \tag{25}$$

We thought if the evaluation was good enough, if it maximized the distance the salamander traveled, it should also satisfy the constraints about the coordination between body and legs, the smoothness of the transitions... Actually it was not exactly like that, as we explain in subsubsection 3.4.1.

Population creation and replacement Every generation of salamanders is composed of 100 individuals, and we simulate 20 generations to complete the evolution of the population. We create the first generation of salamanders by choosing randomly the values of the parameters of the genome, within the following intervals:

- a_{next}, b_{next}, a_{side} and b_{side} in $[-1.0, 1.0]$
- E_i in $[0.1, 1.0]$, $0 \leq i \leq 15$
- τ in $[0.1, 1.0]$

Finally, our replacement strategy is to insert the new individuals from crossover and mutation all together, once they all have been generated (that is, our algorithm is generational rather than stationary). In addition to that, parents may survive if they are good enough still after the children have been generated (in other words, the best 100 individuals, parents or children, are selected for the next generation).

Crossover and mutation The crossover operator we use is this one. Two individuals are selected, whose probability of being selected is a linear function of the rank they have in the population, and are crossed to generate two children as follows: one point is randomly selected in the subarray composed of the parameters a_{next}, b_{next}, a_{side} and b_{side}, in that order. The father's values on the left of that point go to the first child, the values at the right to the second child. With the same point as for the father, the mother's values on the left go to the second child, and the values on the right to the first child. A similar process is repeated for the subarray composed of $E_0, \ldots, E_{15}$, although this time two points are chosen instead of one. The father's τ goes to the first child, while the mother's goes to the second one. Every selected pair of individuals is crossed with probability 0.7.

Figure 14 illustrates the procedure just described.

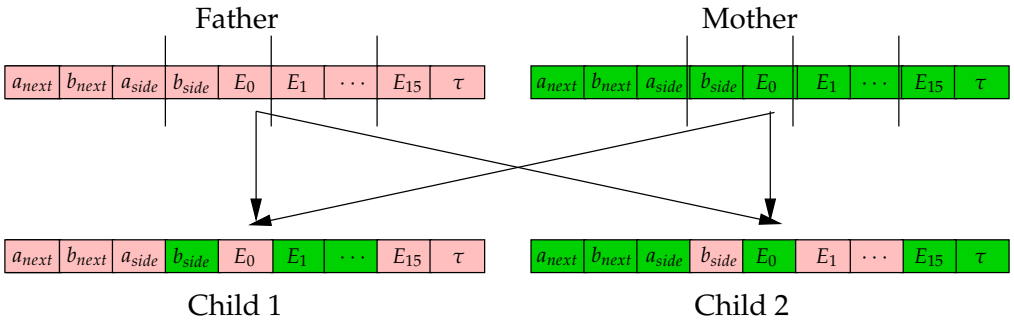

FIG. 14: Example of the crossover operator

Concerning the mutation operator, it works by mutating every individual of each generation with probability 0.2. The mutation consists on the random change of some of the genes of an individual's genome to one of the allowed values for each gene. Every gene is changed with probability 0.2.

3.4.1 Results of the Genetic Algorithm

After the simulation of the swimming gait for the 20 generations of salamanders, we got a maximum fitness of 29.68 meters for the genome of Table 1. The summary of the evolution process can be seen on Figure 15.

a_{next}	0.435352	E_7	0.920918
b_{next}	0.927815	E_8	0.432160
a_{side}	0.947044	E_9	0.482255
b_{side}	0.131792	E_{10}	0.376165
E_0	0.624396	E_{11}	0.096646
E_1	0.446803	E_{12}	0.112944
E_2	0.132297	E_{13}	0.383013
E_3	0.127319	E_{14}	0.533381
E_4	0.531953	E_{15}	0.544814
E_5	0.534632	τ	0.118508
E_6	0.645920		

TABLE 1: Best individual found by the genetic algorithm

With this evolution we got a very fast swimming salamander. However, it swam backwards. To correct this problem, we tried inverting the sign of a_{next} and the order of the E_i parameters as suggested by Yvan Bourquin, but it was not good for us because the salamander robot is not symmetrical. The static parts which do not move—but do have oscillators associated—make this inversion does not produce the desired result. With these changes, the salamander did not follow a straight line and sank sometimes.

In addition to this problem, there was the fact that these parameters were good only for swimming, but we did not know if they were also for walking. So we tried to add the oscillators for the legs (see subsection 3.5 for further details) and get a stable walking gait, at the same time we mantained the parameters we got for swimming. With this configuration, the salamander could not walk very well, probably because the optimum parameters for swimming are not the optimum for walking too. Besides, a mixed simulation (both swimming and walking in every fitness evaluation), was rather tedious to prepare and did not guarantee good results.

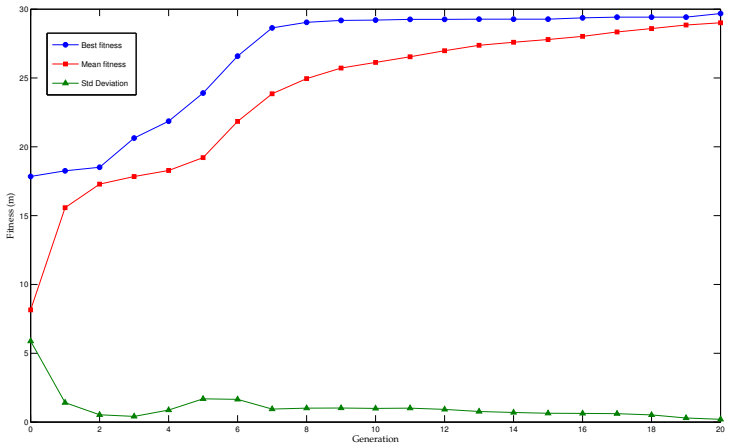

FIG. 15: Summary of the evolution process

Morever, the phase differences of the differents parts of the body of the salamander did not correspond with the ones of the real salamander, neither for walking or swimming.

For all these reasons we decided to adjust all the parameters by hand, which finally did provide quite good results.

3.5 Setting the Parameters: Systematic Search by Hand

Due to the bad results of the genetic algorithm we decided to tune the parameters by hand. It was not a blind search anyway, because we knew the way to obtain arbitrary phase differences between two coupled CE oscillators. Yvan Bourquin commented in an email:

> "The phase shift will depend on the coupling parameters a_{ji} and b_{ji}. Assume you have two oscillators, 1 and 2, and a unidirectional coupling a_{21} and b_{21} from oscillator 1 to oscillator 2. It is possible to obtain any phase shift between the oscillators by tuning the a_{21}/b_{21} ratio. For example with $a_{21} = 0$ and $b_{21} > 0$ you will obtain a zero-phase shift. With $a_{21} = 0$ and $b_{21} < 0$ you will obtain a 180° phase shift. A good approximation of the phase shift you obtain is `atan2(a21, b21)`."

We wished to obtain a set of parameters that allowed to switch between walking and swimming without changing the couplings between the oscillators, in order to be have a more bio-inspired solution.

But before we had to have oscillators to control the legs and their corresponding couplings. The schema we followed can be seen in Figure 16. There is an oscillator for each leg, which is bidirectionally coupled with the oscillator that controls the leg on the other side, and also unidirectionally coupled with all the oscillators of its same side in the trunk or in the tail, if the oscillator controls the front or hind legs, respectively. The couplings between the oscillators in the body are bidirectional again.

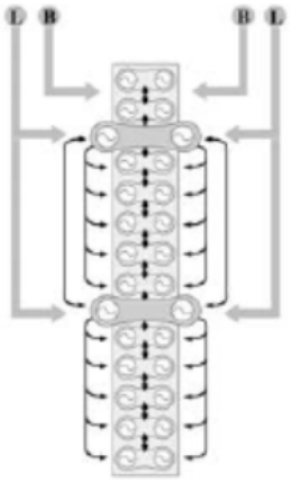

FIG. 16: Structure of the couplings used to control the body and legs of the salamander. (Drawing taken from [9]).

In fact our couplings are slightly different from the ones on Figure 16, because we have one pair of oscillators for the head, three pairs for the trunk and three pairs for the tail.

So this time we removed the *dead* oscillators that were associated with the static parts and included in their place two pairs of oscillators to control the legs.

Control of direction Another interesting feature that we wanted to add to our salamander was the possibility to change its direction, in the water and on the ground. For this purpose, we modify the oscillation of the oscillators by increasing the amplitude (the variable E) of the oscillators on one side and decreasing the amplitude of the ones on the other side. For example, when the salamander turns left, the amplitude of the oscillators on the left is decreased while the one of the others is increased. The more the difference of amplitude is, the more the salamander turns.

3.5.1 Results of the Systematic Search

During the systematic search by hand, the best parameters we found, which nearly provided the perfect phase differences between the several parts of the body and also between the legs, for both swimming and walking, and that at the same time allowed to switch between swimming and walking without changing the couplings (it was necessary to *activate* the leg oscillators only) were the ones indicated on table Table 2.[6]

[6] In the table, *fh* means *front-hind legs*, while *lb* means *legs-body*.

a_{next}	-0.4	a_{fh}	0
b_{next}	0.4	b_{fh}	-0.8
a_{side}	0	a_{lb}	0.8
b_{side}	-0.5	b_{lb}	0.8

TABLE 2: Best individual found by the genetic algorithm

For these simulations, we fixed $\alpha = 1.0$ again and $\tau = 0.15$. We did not have enough time to optimize the E parameters, so we set them all to 0.05, which resulted to be a good value (the oscillators for the legs become inactive when the salamander enters into the water, for which the E values associated to the leg oscillators are set to 0.005, which reduces the influence of these oscillators over the motion of the whole body to practically zero).

With this configuration, we got walking and swimming gaits that practically fulfilled the contraints about the phase differences. That is:

Walking: same phase for the parts in the trunk among them and same phase for the parts in the tail among them too, and counter phase between the parts in the trunk and the ones in the tail.

Swimming: constant difference of phase among all the parts of the body, being the phase of the head the same as the last part of the tail (or, in other words, the body has the shape of a whole period of a sinusoidal function).

Moreover, the legs move in coordination with the body to maximise the speed.

Figure 17 shows some snapshots of the salamander while walking and swimming. In Figure 18 and the phase differences among the different parts of the body can be seen, for walking and swimming. They are not perfect but with the parameters no switching between different couplings is needed. Finally, the transition walking-swimming and swimming-walking is showed in Figure 19, where the smoothness of the process can be appreciated, in opposition to what happened with the sine-based controller.

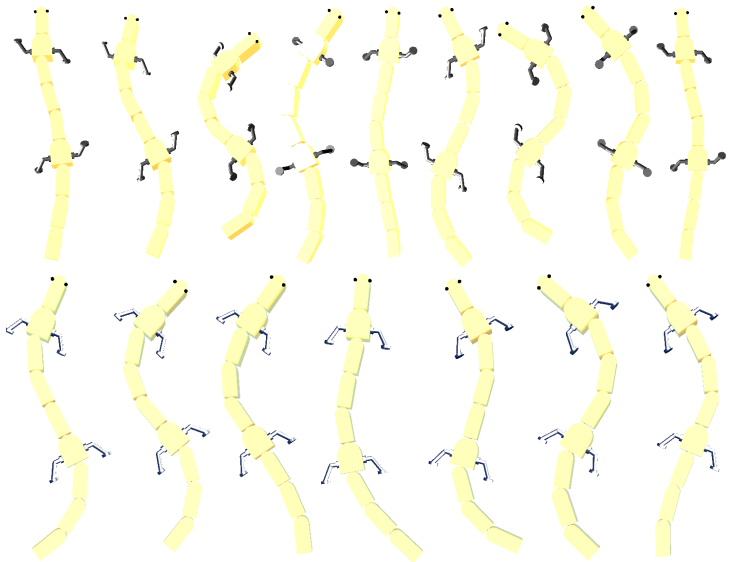

FIG. 17: Sequence of movements of the salamander while walking (up) and swimming (down)

4 Visual System

The second part of the project involves a very simple model of the salamander's visual system. This system should be able to explain and reproduce some of the mechanisms involved when the salamander

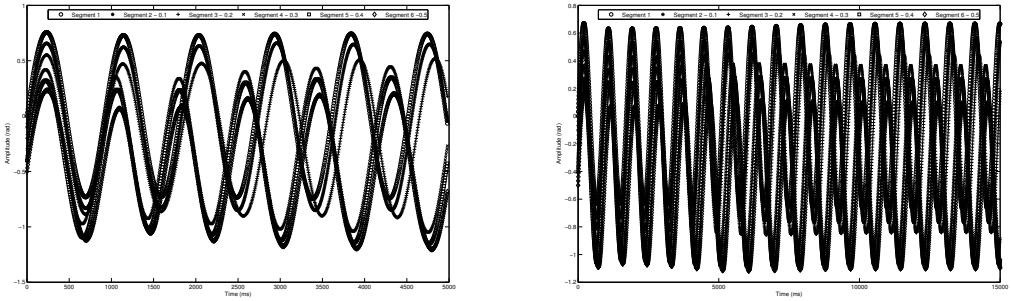

FIG. 18: Phase plot for the 6 moving segments of the body of the salamander for the walking gait (left) and swimming gait (right)

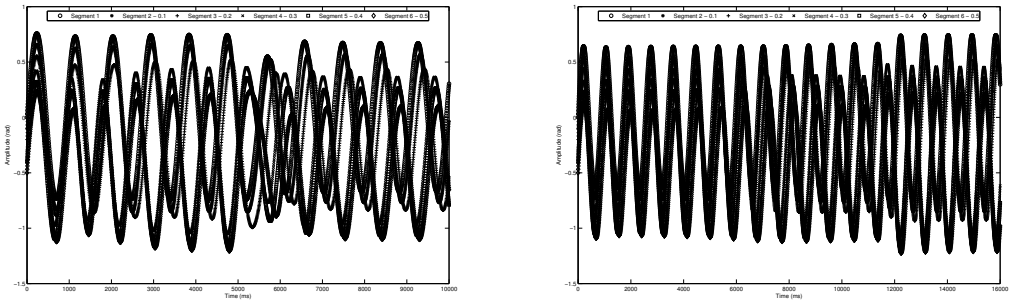

FIG. 19: Phase plot for the transition walking-swimming (left, the transition is produced at time $t = 5.190$ s), and swimming-walking (right, the transition is produced at time $t = 11.360$ s)

is tracking a fly, for instance. In our implementation, the purpose of visual system will be to output a value indicating whether the motor system should turn left or right or go straight on in order to get as close to the fly as possible.

We start with a simple model of the retina, then proceed similarly to the optical tectum and to the brain stem. We finally show what is the final function deciding whether or not to turn, and in which direction.

4.1 Description of the Model

The initial input of the visual system will be two matrices of pixels representing the raw data coming in from the rods and cones, before it is processed by the retina. We don't differentiate between rods and cones. We further assume that, like for man, we can divide the incoming data into three disjoint subsets:

1. Monocular vision describing the left part of the visual field;
2. Binocular vision describing the middle part of the visual field;
3. Monocular vision describing the right part of the visual field.

The cameras on the salamander robots are implemented such that the right half of the left input and the left half of the right input form the binocular vision—they describe the same part of the world on front of the salamander, although the angle is different, see Figure 20.

In order to simplify processing, the color data is converted to grayscale before reaching the retina. Moreover, not the grayscale value of the current image is passed to the retina, but the absolute difference between two consecutive images according to time discretization in the simulation. This is the first step in trying to detect movement from the cameras. We thus have:

$$retina\_input(i,t) = |pixel(i,t) - pixel(i,t-1)| \tag{26}$$

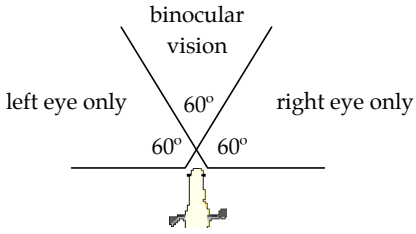

FIG. 20: Schematic view of the cameras' ranges

(For the sake of readability, we reference pixels and cells in matrices with a single index i, with the mapping $i = (x-1) \cdot matrix\_width + y$ if x and y are the coordinates of the current pixel/cell.)

As a matter of fact, this trick of taking the absolute difference between two time steps is not unmotivated by biology. The salamander robot has two on-board cameras of a resolution of 128×64 pixels; in our implementation, both the current and the previous grayscale image are fully kept in memory in order to be able to compute the wanted absolute difference at each time step. This yields a memory requirement for the vision data before the retina of $2 \times (128 \times 64) = 16$ KB per eye if each grayscale value is encoded as an 8-bit `unsigned char`.

4.1.1 Retina

Each retina is represented as a matrix of cells, each of which has a topological map corresponding to a $k \times k$ pixel rectangle of the original grayscale input. All cells have a receptive field of the same size, shifted by p pixels with respect to its neighbours. The activity of retina cell i is generally determined by:

$$retina\_output(i,t) = f_r\left(\sum_{j=1}^{k^2} w_j \cdot retina\_input(rf_r(i,j),t)\right) \tag{27}$$

with $rf_r(i,j)$ referencing the jth pixel of the receptive field of the ith cell of the retina.

A simple possibility—that we implemented—is to set $f_r(h) = h/k^2$ and $w_j = c = 1$, thus computing for each retina cell an activation value equal to the average gray level of its receptive field; i.e., of the average movement in a certain region in front of the salamander. (Note that the left retina is exclusively connected to the left camera and the right retina to the right camera.)

Depending on the values of p and k, which are model parameters, several situations are possible:

- $p \geq k$: no overlapping occurs in the receptive fields. If moreover $p > k$, some values of the input pixels are lost because not taken into account by the retina.
- $p < k$: the respective receptive fields overlap. The resulting retina matrix may look less "sharp," but this technique indeed helps canceling out noise in the original input. In the special case where $p < k/2$, more than two adjacent retina cells will take into account a given input pixel. Generally, $k/(n+1) \leq p < k/n$ implies that each pixel (except those on the border) will be "seen" by $n+1$ retina cells.

The case $k = 4$, $p = 3$ is illustrated in Figure 21. Note that fixing p and k uniquely determines the size of the retina, if the size of the input pixel matrix is known. In our implementation, we chose $k = 4$, $p = 2$ so that every pixel (except those on the border) is in the receptive field of two retina neurons. The size of the retina for an eye is thus 63×31 pixels, yielding a memory requirement of 1953 B per eye.

4.1.2 Optic Tectum

The full optic tectum is also made of a left and a right part. Most generally, it receives a contralateral and an ipsilateral input. To keep things simple, we have only included the contralateral connections in this model, which means that the left part of the tectum is connected to the right retina, and vice-versa.

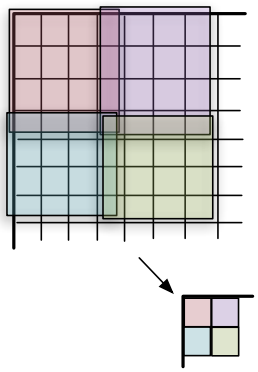

FIG. 21: Cell reduction operation

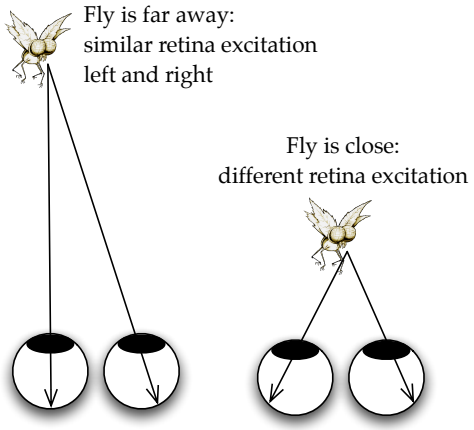

FIG. 22: Finding out how far away an object is

What happens here between the retina and the tectum is a simple cell reduction very similar to the one described by Equation 27. We actually have two additional parameters l and q tuning how this cell reduction is performed, and a new activity function f_t, which correspond to k, p and f_r, respectively. We thus have:

$$tectum\_output(i,t) = f_t\left(\sum_{j=1}^{l^2} w_j \cdot retina\_output(rf_t(i,j),t)\right) \tag{28}$$

with $rf_t(i,j)$ referencing the jth retina cell of the receptive field of ith cell of the tectum.

The same considerations on l, q and f_t apply as those on k, p and f_r in subsubsection 4.1.1. The tectum matrix now gives a rough view of where things are moving in front of the salamander; it becomes rougher as l and k increases (which also reduces the number of cells in the retina and tectum if the initial input size is fixed) but also helps canceling out noise. In our implementation, we used $l = k = 4$ and $q = p = 2$ and applied the transfer function $f_t(h) = h/l^2$ with $w_j = c = 1$. The size of the optic tectum for an eye is thus 30×15 pixels, yielding a memory requirement of 450 B per eye.

At this point, the left vision is summed up in the right part of the tectum and vice-versa; we haven't used any information about monocular or binocular vision yet. This happens in the brain stem.

4.1.3 Brain Stem

In our model, the brain stem will theoretically be responsible for differenciating movement happening far away from the salamander from objects moving closer to it. Now, the binocular vision will be used exclusively.

The idea behind this brain stem model is that objects that are far away will produce a very similar excitation in the right and left retinas, as opposed to close objects which activate more distant areas (this is showed in Figure 22). Thus, taking the absolute difference between the two activation matrices left and right should filter out distant objects.

This can only be done with binocular vision: we now need data from both parts of the tectum. This corresponds to the right half-matrix of the right part of the tectum (actually containing data corresponding to the right part of the left camera) and to the left half-matrix of the left part of the tectum. Taking the absolute difference between them leads now to the equations of the brain stem (using x, y coordinates and with w_t the width of the tectum, in cells):

$$brain\_stem\_output(x,y,t) = |tectum\_output_{left}(x,y,t) - tectum\_output_{right}(w_t/2 + x, y, t)| \tag{29}$$

In our implementation, the brain stem has a size of 15×15 pixels, which correspond to a memory requirement of 225 B.

The brain stem should now display high activation values for close objects moving rapidly. If we further divide the brain stem matrix into a left and a right part and sum over these half-matrix, we obtain two values, which we call *end activity left* and *right* ($w_b = w_t/2$ is the width of the brain stem):

$$ea_l(t) = \sum_{x \leq w_b/2} \sum_y brain\_stem\_output(x, y, t) \tag{30}$$

$$ea_r(t) = \sum_{x > w_b/2} \sum_y brain\_stem\_output(x, y, t) \tag{31}$$

This, however, still ignores half of the total data, namely all monocular data. We define the end activity for the monocular vision as follows:

$$ea_{ll}(t) = \sum_{x \leq w_t/2} \sum_y tectum\_output_{right}(x, y, t) \tag{32}$$

$$ea_{rr}(t) = \sum_{x > w_t/2} \sum_y tectum\_output_{left}(x, y, t) \tag{33}$$

The whole model of the visual system can be briefly summed up by Figure 23. The four variables ea_{ll}, ea_l, ea_r and ea_{rr} are the output of the visual system and will be used, with various weights, in order to control direction.

The total memory requirement for our implementation of the visual system (including the 4 final output values; not including the color output of the cameras automatically handled by Webots) is 37803 B.

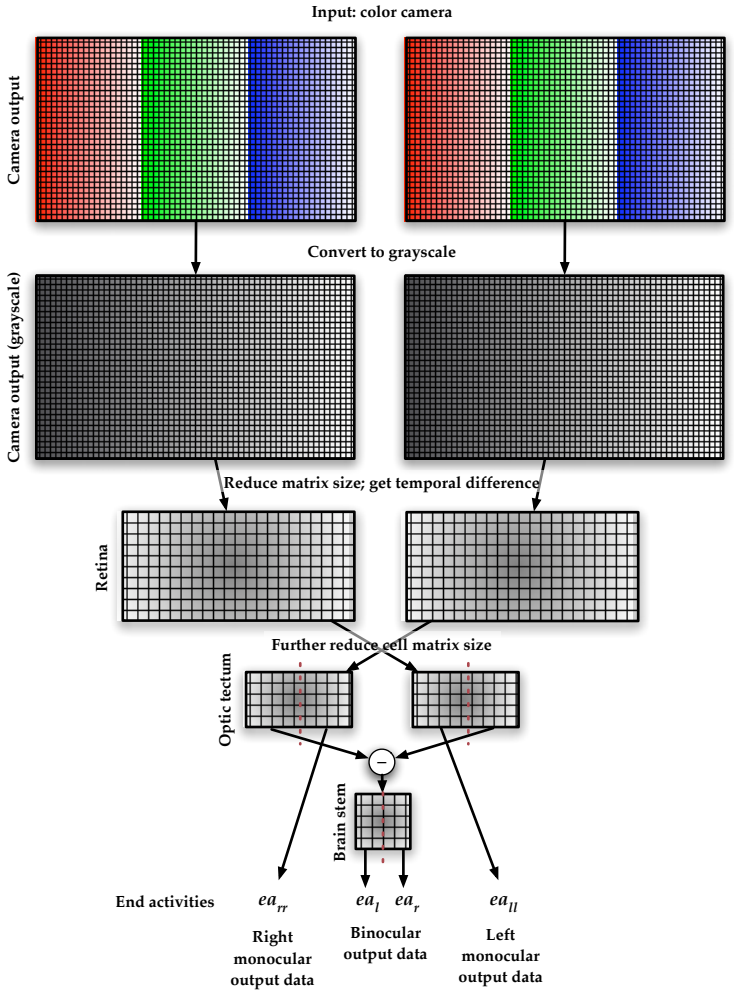

FIG. 23: Schematic view of the full model of the visual system

4.2 Tests: Tracking the Fly

The strategy we use to make the salamander track the fly is quite simple. First the salamander has to decide whether the fly is on the left, on the right, just ahead or if it does not see the fly. This is accomplished by the following rules, checked in the order they are presented:

1. If the activity in the right part of the tectum is higher than the activity in the left part and there is no activity in the left part of the brain stem, then the fly is on the right.
2. If the activity in the left part of the tectum is higher than the activity in the right part and there is no activity in the right part of the brain stem, then the fly is on the left.
3. If there is activity in both parts of the brain stem, left and right, then the fly is more or less in front the salamander; we treat this case as if the salamander could not see the fly (no activation altogether).

Then, once the salamander has decided where the fly is, it turns in the direction the fly is. If the salamaner cannot see the fly, it checks the last time it saw it: if that moment was more than two seconds ago, then the salamander starts going straight; otherwise it keeps going in the same direction because the oscillation of the head makes the salamander lose the fly for an instant although the fly does not change it position.

This simple approach resulted to be quite effective when tracking the fly. In most of the simulations we did, the salamander was able to track the fly for more than one minute, in the water and on the ground.[7]

5 Conclusion

In this report we presented the model of the salamander's locomotion and visual systems to control a simulated salamander robot. First, we studied how to perform locomotion with a very simple, naive sine-based controller, we optimsed it to maximise the speed of the salamander and we showed its advantages and drawbacks. Then, we explored a more sophisticated, biologically-inspired controller based on central pattern generators and nonlinear oscillators, which reported much more benefits than the previous approach. Finally, we built up a visual system, also taking as source of inspiration the visual system of the real salamander, and tested it by tracking a simulated fly robot.

[7] We used for that purpose the worlds `infine_water_with_fly.wbt` and `infinite_ground_with_fly.wbt`.

References

[1] The Biologically-Inspired Robotics Group at the EPFL. Salamander locomotion files: http://birg.epfl.ch/page28707.html. Animations, movies and papers.

[2] J. Buchli and A.J. Ijspeert. A simple, adaptive locomotion toy-system. In *From Animals to Animats 8. Proceedings of the Eighth International Conference on the Simulation of Adaptive Behavior (SAB'04)*, pages 153–162. MIT Press, 2004.

[3] W. Gerstner and W. Kistler. *Spiking Neuron Models: An Introduction*. Cambridge University Press, New York, NY, USA, 2002.

[4] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[5] T. Graham Brown. The intrinsic factors in the act of progression in the mammal. In *Proceedings of the Royal Society of London*, pages 308–319, 1911.

[6] S. Grillner, T. Degliana, Ö. Ekeberg, A. El Marina, A. Lansner, G.N. Orlovsky, and P. Wallén. Neural networks that co-ordinate locomotion and body orientation in lamprey. *Trends in Neuroscience*, 18(6):270–279, 1995.

[7] A.J. Ijspeert. Design of artificial neural oscillatory circuits for the control of lamprey- and salamander-like locomotion using evolutionary algorithms, 1998. PhD Thesis.

[8] A.J. Ijspeert. A connectionist central pattern generator for the aquatic and terrestrial gaits of a simulated salamander. *Biological Cybernetics*, 84(5):331–348, 2001.

[9] A.J. Ijspeert, A. Crespi, and J.M. Cabelguen. Simulation and robotics studies of salamander locomotion. Applying neurobiological principles to the control of locomotion in robots. *Neuroinformatics*, 3(3):171–196, 2005.

[10] A.J. Ijspeert, J. Hallam, and D. Willshaw. From lampreys to salamanders: evolving neural controllers for swimming and walking. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S.W. Wilson, editors, *From Animals to Animats, Proceedings of the Fifth International Conference of The Society for Adaptive Behavior (SAB98)*, pages 390–399. MIT Press, 1998.

[11] O. Michel. Webots: Professional mobile robot simulation. *Journal of Advanced Robotics Systems*, 1(1):39–42, 2004.

[12] C.S. Sherrington. Flexion-reflex of the limb, crossed extension reflex, and reflex stepping and standing. *Journal of Physiology*, 40:28–121, 1910.

[13] Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.