BioTrack: Computing with Living Components

Peter Beyls
The School of Arts,
University College Ghent, Ghent. Belgium
www.kask.be
peter@peterbeyls.net

Abstract

We report on BioTrack, a real-time audiovisual installation expressing interest in behavioral complexity observed in nature. Specifically, small creatures evolve and engage in a computer-controlled habitat - sensors, activators and software afford conditioning and tracking creatures' activity as situated in a small glass container. Movement is detected by means of a camera and custom computer-vision software. Variable light patterns are sent into the container using a 3D printed structure holding 16 LEDs (light emitting diodes). A machine learning algorithm targets to maximize behavioral diversity by optimizing the relationship between specific light patterns and the complexity of observed trajectories in space. Evolving trajectories reflect in the way sound patterns develop over time. Preliminary experiments reveal noisy yet non-trivial results

1. Contextual note

In a seminal 1974 performance, *I Like America And America Likes Me*, Joseph Beuys shares a small room with a live coyote for three days. Decidedly relevant given the subject of this paper, the artist and the animal managed to develop a functional relationship of co-habitation based on mutual understanding and appreciation. The coyote's behavior evolved from aggressive and cautious to approachable and sociable. Actual evolving behavior is close to the heart of this paper as it considers software conditioning of small biological creatures.

Unlike Damien Hirst's tiger shark preserved in formaldehyde, the anthropopathic robots created by Chico MacMurtrie are prime examples of essentially hand-crafted structures nevertheless suggesting deeply poetic and organic life-like qualities [6].

It is often claimed that 'art imitates life', however, in recent years, the substituting components of life itself have become not merely subject matter but actual material constituents towards the construction of artefacts. A wide range of ideas, methods and technologies from the biologists laboratory have been appropriated by the artist also raising questions of social and ethic implication [14].

Consider, Alba, a fluorescent rabbit, a trans-genetic experiment by Eduardo Kac (2000), created by genetic engineering rabbit DNA, specifically through synthetic mutation of the green fluorescent gene found in the jellyfish. Alba is a prominent example of generative art beyond computing – the synthesis of structure through explicit instruction without resorting to software. Consider, French performance artist Orlan, viewing her body as software subject to perpetual modification.

Oron Catts, leading artist in the Tissue, Culture and Art Project (TC&A), has been designing structures merging living biological material and synthetic constituents since 1996 suggesting that “the body cannot survive without organs and cells, but the latter two groups can survive without body” [3]. TC&A develops biological art collaborations and advocates the notion of semi-living entities; cells and tissues isolated from organisms and coerced to grow in predetermined shapes. Such research implicitly comments on the human condition and raises imperative philosophical and ethical questions.

Interfacing art and biology is by no means a recent phenomenon. Early experiments in art and technology either took inspiration from interesting behavior observed in biological workspaces to inform the creation of sculptural installations exhibiting life-like qualities [5, 9] or evolving virtual creatures in software [10]. Still others choose to fully integrate living creatures in their work.

SEEK is a fine example of an early project involving live animals developed by Nicholas Negroponte and his team at MIT, on display at the NY Jewish Museum in 1970 [4]. SEEK is designed as a sensor/activator system; software senses a 3D environment composed of small blocks, evaluates changes and reorganizes the world accordingly with live gerbils moving inside this kind of variable architecture. Remarkably, the software chooses to either amplify the effect gerbil actions or to compensate i.e. reorganize the world according to a specific design. Note: SEEK implements a closed loop system interfacing biological and synthetic behavior, effectively suggesting a complex hybrid biotope displaying unpredictable though coherent behavior.

A single cell bio-electronic component may support hybrid hardware-wetware computing: sound can be sent into a Petri dish holding slime mould (*Physarum polycephalum*), in turn emitting electrical signals to be translated back into sound – biological behavior becomes a mapping interface in interactive computer music performance [8].

My project *Fishbowl* [1] addresses collective behavior and fish swimming patterns. A reinforcement learning algorithm aims to optimize behavioral diversity by influencing performance through external stimulation operating light patterns. Computer-vision tracks global flow i.e. intensity and direction of the centroid of movement. The basic idea is to maximize flow signal diversity, a process echoing in the sounds produced through sonification.

DeepLabCut is an open-source tool developed at Harvard University, it explores deep learning for training neural nets to track animal postures and behavior. Tracking is deemed attractive since motion offers an impression of the development of intentionality in the brain [7].

2. Introduction to BioTrack

BioTrack is an experimental art-research project interfacing biological and synthetic communication. Highly speculative, it suggests the exploration of spontaneous behavior of natural creatures in interaction with machine-generated information and how internally motivated, innate behavior might be influenced by external audiovisual stimuli. Specifically, we are interested in the unfolding locomotive complexity of biological creatures (such as ants) situated in a small, isolated biotope, typically a 15 cm Petri dish.

Spatial behavior of creatures is tracked in 2D space using computer-vison; movement information is extracted as tiny ‘trajectories’ reflecting short-term history dynamics. Trajectory analysis yields information on the *quantity* and *quality* of the physical activity, respectively, how much activity is taking place, if any, and how complex that activity actually is. In other words, levels of amplitude and levels of interestingness implied in the data is made explicit.

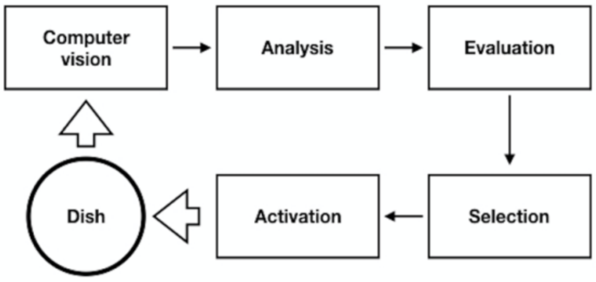


Fig. 1 Schematic overview

Following the design principle of maximization of diversity [13], we aim to maximize the complexity of the natural activity inside the dish using a reinforcement learning algorithm. Since optimization is highly inhibited by internal and external constraints, ultimately, wave-like activity will probably develop reflecting the process of continuous adaptation.

We might interpret the trajectories as control signals issuing from a hybrid bio-machine structure and view them as a control structure exercising parametric control over a sound-generating algorithm. Then, the manifestation of a process of biological life interfaces actively with a process of cultural design. More exactly, implicit natural behavior becomes a qualitative source of information influencing the development of sound patterns in an algorithm of explicit design. This method clearly avoids the notion of ‘control’ in favor of ‘influence’ – which is an aesthetic choice in the context of evaluating different mapping strategies in a typical computer music application. Mapping exists as a continuous scale between two conceptual opposites: (1) *responsive* systems, where control data selects responses from a palette of pre-designed responsive options and (2) *interactive* systems suggesting a more symbiotic human-machine relationship reflected in unpredictable yet intelligible machine responses [2].

Figure 1 shows circular action in a sensor-action framework; the camera image is analyzed, tracking changes in developmental quantity and quality. A reinforcement learning algorithm selects policies (hardware activation patterns) aiming to optimize future behavioral diversity in the dish.

Figure 2 shows the computer-vision GUI with current camera image, a single tiny creature, its moving contour (red color) obtained through background-subtraction, a number of sliders for parameter tuning and radio buttons to select computer-vision functions. Visualization also includes flow sensor grid activation (light blue dots), current centroid of movement (white circle) and trajectory (green).

Hardware

Two different sized prototypes of hardware interfaces were built aiming to accommodate distinctive sized action spaces. The first interface (figure 3) features a 5.5 cm Petri dish, 8 LEDs (light emitting diodes) used as activators and 8 LDRs (light sensitive resistors) used as sensors. The LDRs were removed from the second much larger design which includes a 17 cm glass container, 16

LEDs and one piezo element used as a global activator while sensing relies exclusively on computer-vision software. All components interface with an Arduino board running the Standard Firmata library and custom software.

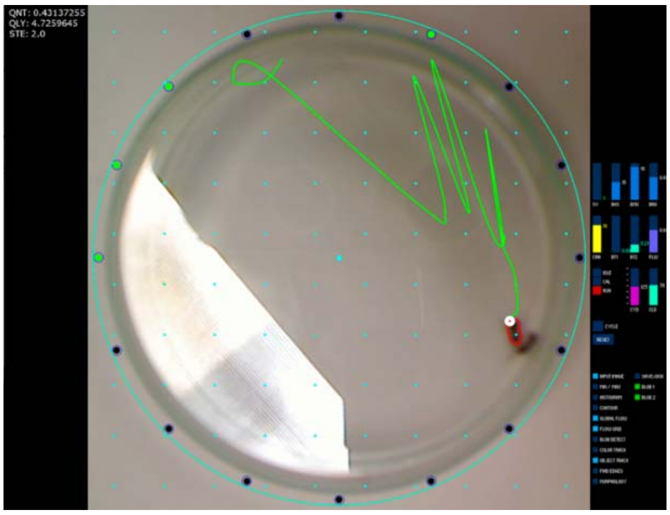


Fig 2: Main interface of BioTrack application

3. Implementation

3.1 Analysis functions

BioTrack is organized as an object-oriented system written in Processing [12] with extensive use of external libraries, including functionality for Supercollider, Arduino, OpenCV and OSC (Open Sound Control).

Software serving behavioral analysis is based on OpenCV. Every single module addresses a particular aspect of image complexity while the user may select an arbitrary collection of modules to run in parallel as to compute an overall image analysis result in real-time. However, in practice, some algorithms are computationally expensive affecting global performance. In particular, two computer-vision algorithms are helpful in developing a high-level interpretation of creature(s) activity as reflected in consecutive image frames: flow detection and background-subtraction.

A *flow* algorithm computes both the local and the global flow i.e. the strength and the direction of change in the image – the data is reflected in the size and angle of a vector. In addition, a grid of virtual flow sensors overlays the actual camera image, a clear impression results of specific location and strength of physical activity in the dish. Every process cycle yields a list of flow data; heading and intensity of change – the global *quantity* and *quality* levels are computed from the data: quality depicts the diversity of angles (the length of the list vs. the number of unique angles) while quantity is computed by averaging the length of all flow vectors in the list. However, when dealing with tiny creatures (such as ants or small spiders), we observed that flow data is too weak, for that reason we turn to background-subtraction.

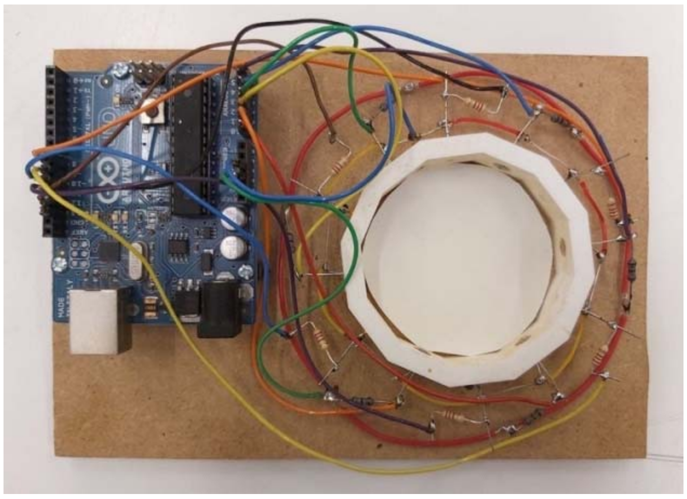


Fig 3: Early prototype with Arduino board

A *background-subtraction* algorithm continuously compares consecutive image frames generating a collection of contours suggesting both an impression of the volume and diversity/complexity of change. Background-subtraction tracks the amount of change (total number of contour points and total contour surface) and the centroid of change (the averaged global center origin of all contours).

Background-subtraction aims to extract *trajectories of change* from the analysis of consecutive frames; a trajectory is an abstraction for a 'path' in 2D space – we are interested in the complexity of the path and how it changes over time as a consequence of external stimulation e.g. light patterns and sound. At 12 frames/second, up to 5 trajectories are accommodated taking care of 5 groups of contours – a single trajectory is a 20-elements long FIFO (first in/first out) data structure accepting the respective centroid of every group.

Given a trajectory of points in 2D space, we may compute both the angles and distances between any two points. Considering the *continuity* (deliberating change in consecutive data items) and *diversity* (studying all data; the ratio of the number of data items vs. the number of unique items) we get an impression of the global complexity and eventually, how it progresses inside the history of a single trajectory.

Then, two critical values result: *quality* and *quantity*, respectively. Their evolving values will impact the learning procedure.

3.2 Learning

Optimization is the goal of learning. Our wish to maximize behavioral complexity requires a method to select appropriate, promising stimuli (LED light patterns, and particular sonic activation) to be sent into the dish. However, we have no clue for how effective any given external activation might be; therefore, we suggest an optimization method based on reinforcement learning (RL) [11], more specific, a variation of the Q-learning (QL) algorithm [15] was implemented. RL aims to select the best actions – in any possible system state – in order to maximize the reward. Therefore, RL is considered a method of experience-based unsupervised learning. Q-learning coordinates action selection: locally, select the optimal action given a state and globally, learn a policy that maximizes the *total* reward. QL typically tabulates state-action pairs and q-values (equivalent to efficiency), in the context of the present project; estimated proficiency to maximize diversity.

Initially, a collection of random policies is computed and applied within a finite process cycle (typically 10 seconds) and evaluated – the efficiency/reward being proportional to the strength of the interval in behavioral complexity between beginning to end. At the start of the RL process, we are in the *exploration* phase; random selection and evaluation of policies. After a while, we gain some understanding of potentially effective policies and enter the *exploitation* phase; moving on from exploration and discovery to selection of the action featuring the highest q-value.

Implementation of learning

Computing the present quantity and quality (QQ) values follows a self-regulating sensitivity windowing algorithm; the minimum and maximum sample window edges move up and down trying to (1) accommodate outliers through expansion as well as maximizing sensitivity through gradual compression.

The concept of a system *State* is crucial; QQ values (0 ~1.0) are mapped to 0 ~ 3, consequently combining the data yields a State between 0 and 15.

Learning manages the list *saPairs*: a collection of State-Action pairs (SAP), a single SAP holds a vector specifying the fitness of every potential action given one particular state. At any time, the number of possible actions equals 16. Learning proceeds as follows:

- The new State is computed form the present QQ values.

  newState = computeStateAdaptive(quality, quantity);

- The reward/punishment is computed from the signed interval: current quality minus the previous quality.

  reward = (currentQuality – previousQuality).

- Get the efficiency (q-value) after using the last action in the previous state. Therefore, check for an existing SAP holding that State. If it exists, return the q-value of that action, if not return 0; i.e. that State has never occurred before.

  thisQ = getQValue(previousState, lastAction);

- Look for an existing SAP featuring the new state.

  StateActionPair p = getSAP(newState);

- If *p* equals null (signaling a newly observed state), a new SAP is created and appended to the *saPairs* list:

  StateActionPair sap = new StateActionPair(newState);
  *newQ = (float) thisQ + learningRate * (reward - thisQ);*
  sap.qValues[lastAction] = newQ;

- If *p* is not null (signaling the SAP already exists), collect the highest value in the q-values vector and compute the new q-value and update the q-value for the lastAction in the given SAP.

  maxQ = p.getMaxQValue();

  *newQ = (float) thisQ + learningRate * (reward + maxQ - thisQ);*

  updateQValue(oldState, lastAction, newQ);

- We are now ready to select a next action given the new state and update the previous quality.

  newAction = selectAction_E_GREEDY(newState);
  prevQuality = quality;

- The *selectAction_E_GREEDY* function is conditioned by the *epsilon* (0 ~1.0) parameter, it specifies a probabilistic ratio between exploration and exploitation activity in learning. Exploration selects a random action without relying on learned information. Exploitation entails a selective procedure aiming to select to most promising action based on the evaluation of previous learning cycles. However, we first check the qValues for the current state;

  qValues = getQValues(state);

  if (random(1.0) < epsilon || qValues == null) {

  action = int(random(nrActions)); // exploration

```
} else {
          // exploitation
          find maxQ in qValues
          create list of all actions
contributing to maxQ
          randomly select and
return action from the list
}
```

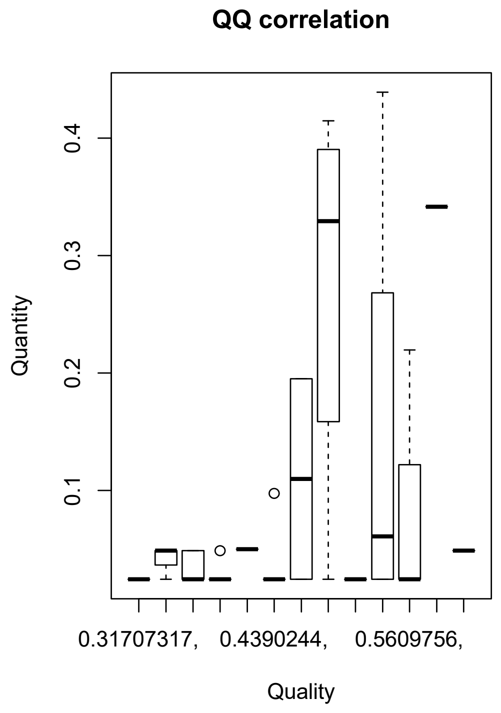
Learning is a gradual process of optimization requiring sufficient cycles before the influence of external conditioning becomes effective. In particular, when interlocking living creatures in a computational process, estimation of the required learning time seems impossible. More systematic experiments are definitely needed. Some preliminary experiments are documented next.

4. Experiments

Short experiments run for 45 cycles, 40 samples per cycle and sampling interval of 200 msec.

Box plots in figure 4 suggest, given low quality values, a strong correlation to exist between quantity samples, with higher quantity values, the agreement fades. Globally, the quantity median values show a bell-like distribution – both very low and very high-quality levels correspond to low-quantity. These findings are contradicted in experiment 2 as quantity values are widely distributed, presented in figure 5.

Graphs for both experiments as shown in figures 6 and 7, clearly reveal a gradual decline in action-frequency – explained by the gradual progression from exclusive exploration selection to maximum exploitation selection in the learning algorithm. In experiment 1 (figure 6), action frequencies exist as two groups of low and high values. In figure 7, oscillations between action-frequency value 5 and lower values seem to agree with the chaotic pattern of QQ correlation in figure 5.



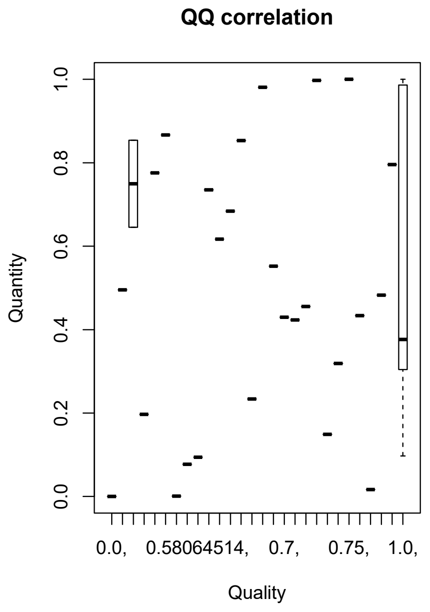
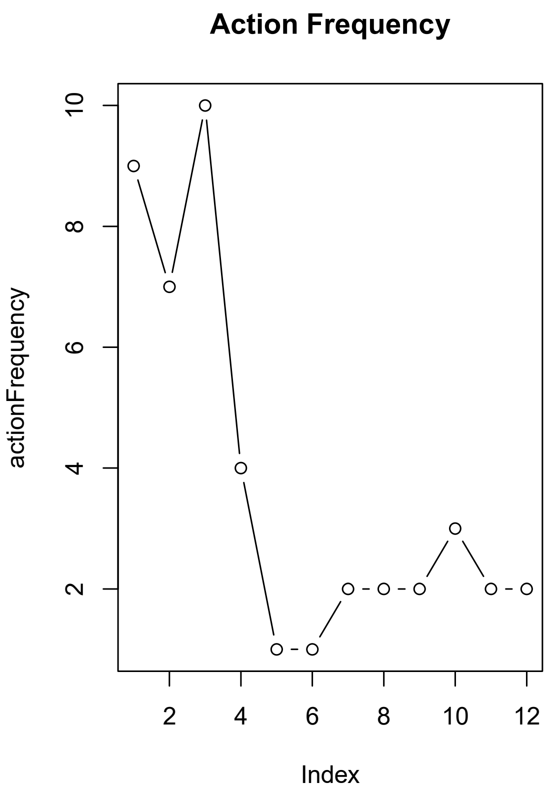


Fig. 4 Experiment 1
Fig. 5 Experiment 2



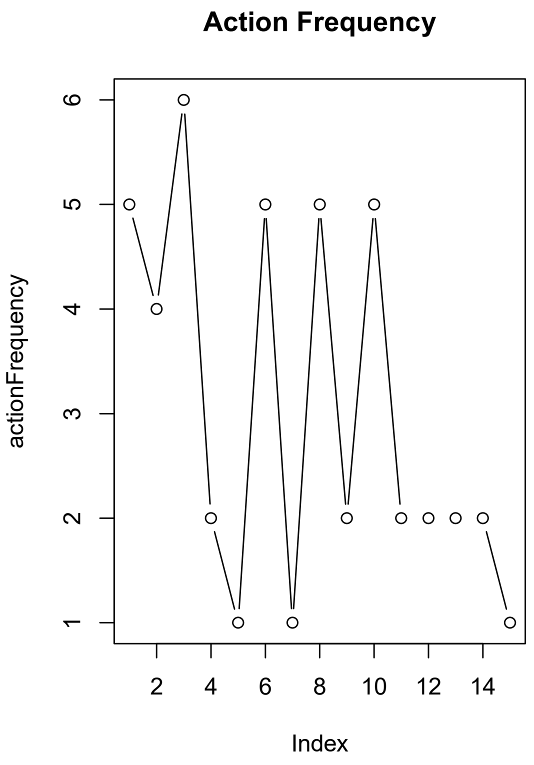


Fig. 6 Experiment 1
Fig. 7 Experiment 2

5. Conclusion

This paper documents preliminary results of BioTrack, an audiovisual experiment exploring behavior of living creatures and how their spatiotemporal activity could be optimized using a reinforcement learning algorithm.

Obviously, experiments of much longer duration are indispensable to fully understand the scope of this speculative project. Perhaps it requires stronger physical conditioning to reduce noisy behavior to a minimum. By definition, the behavioral complexity of living creatures seems impossible to estimate – there are wide gaps of inactivity in the acquired data. The relationship between activation patterns and ensuing locomotion seems unclear. Many unknown biological processes seem to contribute to unfounded noisy behavior. Long term experiments might reduce noise levels and reveal delicate, intricate action patterns.

Acknowledgement
I am grateful to biologist Maria Boto Ordonez, for discussions and creating a unique laboratory structure in support of biology and the arts. In addition, I thank Elias Heuninck for designing and creating the sensor-activator structure using 3D printing.

References

1. Beyls, P (2012) Interaction in Hybrid Spaces, *ISEA2012 Proceedings*, Albuquerque, NM
2. Beyls, P (2019) Art as a Living Interface, *Design, User Experience, and Usability. Design Philosophy and Theory*, HCI2019, Orlando, FL Springer

3. Catts, O and Zurr, I (2006) Towards a New Class of Being: The Extended Body, *Artnodes* 6(2), 1-9
4. *Software. Information technology: its new meaning for art*, The Jewish Museum, New York, 1970
5. Ihnatowicz, E (1976) Towards a Thinking Machine, in: *Artist and Computer*, R. Levitt (ed.), Harmony Books, NY
6. Kammerman, M et. al. (2018) *Machines and Robots*, Edition Digital Culture, Christoph Merian Verlag, Switzerland
7. Kwok, R (2019) Deep learning powers a motion-tracking revolution, *Nature 574*, 137-138 (2019)
8. Miranda, E. R., Braund, E. A Method for Growing Bio-memristors from Slime Mold. *J. Vis. Exp.* (129), e56076, doi:10.3791/56076 (2017)
9. Pask, Gordon (1971) in: *Cybernetics, Art and Ideas*, J. Reichardt (ed.), New York Graphic Society
10. Sims, Karl (1994) Evolving Virtual Creatures, *SIGGRAPH '94 Proceedings*, 319-328
11. Sutton, R and Barto, A (2018) *Reinforcement Learning: An Introduction*, Second Edition, MIT Press, Cambridge, MA
12. Fry, C and Reas, C (2007) *Processing*, MIT Press, Cambridge, MA
13. Ox, Jack (2019) Peter Beyls and Algorithms, in: *Coming Full Circle*, Verbeke Foundation Belgium
14. Anker, S and Nelkin, D (2003) *The Molecular Gaze*, Cold Spring Harbor Laboratory Press, NY
15. Watkins, C and Dayan (1992) P Technical note: Q learning, *Machine Learning*, 8 (3/4), 279-292