Homeostatic robot control using simple neuromodulatory techniques

James C. Finnis

Computer Science, Aberystwyth University, Penglais, Aberystwyth SY23 3DB
jcf1@aber.ac.uk

Abstract. The UESMANN (Uniform Excitatory Switching Multifunction Artificial Neural Network) architecture has been shown to produce interesting transitions between multiple behaviours using an extremely simple neuromodulatory regime. Previous work has concentrated on discrete classification tasks. In this work, three different simple neuromodulatory architectures including UESMANN are used to control a robot in a homeostatic task.
The experiments show that UESMANN produces interesting and useful transitional behaviour in an embodied system, learning the two tasks in the same number of parameters (i.e. network weights) as networks which learned each individual task.

Keywords: Neuromodulation, neural network, backpropagation, robotics, long-term autonomy, homeostasis

1 Introduction

This work compares three methods of modulating the behaviour of a neural network with a single parameter to perform some "blend" between two learned behaviours. Smooth transitions between behaviours may be useful in situations where a system's behaviour should change with environmental conditions, perhaps to achieve homeostasis. In such systems, a small change in the environment (or time) should induce a similarly small change in the behaviour. However, often it can help if this transition is not completely smooth – some complexity in the transition can help by introducting variety into the output, as our experiments on a physical robot show.

The three methods compared are neuromodulatory techniques based on feed-forward artificial neural networks. They are: linear interpolation between the outputs of two networks (output blending), using an additional input to carry the modulator value (h-as-input), and the UESMANN architecture. The latter, introduced in [1], is interesting because it is extremely simple and requires no extra parameters beyond the weights and biases: it is based on multiplying all the weights by a function of the modulator. It may be possible to train more than two functions in a UESMANN network, although this may not converge if

the functions are too dissimilar. The work referenced above has examined the network in pattern recognition problems.

The present work involves generating homeostatic behaviour in a robot using two behaviours learned offline: sonar-based wall-avoidant wandering, and phototropic (simulated) charging. Such homeostatic systems, striking a balance between task performance and exploitation of resources, are useful in long-term autonomous settings[11]. While this is a well understood action selection task, using a real robot with noisy sensors and actuators to perform a neuromodulatory task may bring out important differences in the three methods.

2 UESMANN

In biological systems, the behaviour of a group of neurons may be modulated by neuromodulatory chemicals, typically by acting on the synapses between them[4]. Much work has been done on artificial neuromodulatory systems such as GasNets[3, 13, 6] and Artificial Endocrine Systems[8, 11, 7]. Because CTRNNs can realize any dynamical system[2], CTRNNs may also encompass a neuromodulatory model. CTRNNs and GasNets are typically generated by evolutionary algorithms[5].

UESMANN networks, in contrast, are trained using a variant of backpropagation of errors[10]. They are much simpler, consisting of a single modulator with a single, uniform action across all the weights – thus forming an artificial analogue of a group of neurons acted on by a single modulator.

The UESMANN architecture is a feed-forward network with one or more hidden layers, with a global modulation parameter h. Each unit has the form

$$a_i^l = \sigma\left(b_i^l + (h+1)\sum_j w_{ij}^l a_j^{l-1}\right) \tag{1}$$

and will perform one function at $h = 0$ where the weights take their nominal values, and one at $h = 1$ where the weights are doubled. As such, they are close to the simplest possible form of neuromodulatory ANN. Training the network is currently done by UESMANN-BP, a stochastic hill-climbing technique. This involves presenting alternate examples from each function, with h set to the appropriate value, and backpropagating the errors. Full details are in [1].

3 Methodology

Our aim is to compare the behaviours of different multi-function network architectures in a problem domain with continuous outputs and noisy inputs, in contrast with the discrete classification tasks studied in[1]. Robot control is a difficult problem which often requires switching or blending different behaviours in order to balance various objectives, such as data collection and power management. Robot sensors are also noisy, and actuators often do not respond in an ideal manner.

The target robot is a Pioneer 2DX with eight sonars and an omnidirectional camera as light sensor, and our aim is to produce wandering behaviour with the simulated charge high, and phototaxis with the charge low (i.e. the light is simulating a power source beacon).

Neural network training is not guaranteed to converge to a solution, particularly where one network is required to learn two functions. The initial weights and the training data provided may result in a poor local minimum being found. Therefore a number of networks are trained for each architecture and tested using a simple simulator with perfect differential steering and sensors. Using the simulator allows us to run at a suitably high speed to both generate sufficient training data (different for each network instance) and evaluate the networks over a number of runs. 10 networks were generated for each type, and evaluations done on each network.

The best networks are carried forward into the robot experiments, where they are evaluated at two different levels of charging efficiency and two different initial directions. Five runs for each combination are performed and analysed to determine differences in learning ability and transition behaviour.

4 Simulator experiments

Our very simple simulator runs two rule-based controllers to generate training data. One controller will wander and avoid using sonars, the other will head towards the light and stop at a particular intensity: this is a simple model of an "exploration/exploitation" problem. When generating training data, there is no charge model, just a small probability of the robot switching from one controller to the other (0.001 per tick). The log of inputs, outputs and controller ID comprise the training input for the networks. A new set of training data is generated for each network instance, to avoid any irregularities in an individual training set skewing the result. Each training set consists of 200000 training log entries (recorded every 0.01s of simulated time), and the robot's maximum speed is 1 ms^{-1}. The training environment is an enclosed 12×12m box, with 2 walls extending from the sides interrupting the space. During training, the robot is occasionally randomly turned and repositioned to avoid loops and ensure as many situations are represented in the data as possible. Note that the *light* controller will also stop when the light is bright, to add some complexity to the behaviour and hopefully cause stopping at the light when charge is low. The IDs stored in the log for the controllers are 0 for *sonar* and 1 for *light*. During evaluation, setting the modulator h to 0 should produce *sonar* behaviour, and 1 should produce *light*.

Training the networks is done offline after the data-generating run is complete. Each network is presented with the randomly shuffled examples, with 3×10^7 presentations in total (i.e. each example is presented 150 times). In the case of output blending, the appropriate network is presented with the example; for UESMANN and h-as-input the network is trained with the controller ID as the parameter h.

All networks had 8 sonar inputs, 8 light inputs and 2 motor outputs, with a single hidden layer of 16 nodes. The learning rate $\eta = 0.1$.

4.1 Simulator evaluation

Each trained network is evaluated, this time on a simpler arena without the internal walls: during actual experiments, the arena must be obstacle-free since the *light* controller cannot pathfind (we ignore this during the initial training data generation). 10 runs are performed for each network, each starting at a random position and orientation. The runs end at a simulated 1000s or when the system runs out of charge.

The charge and modulator model used is very simple. The virtual battery has a charge in the range [0,1]. The motor uses power linearly with speed, on top of a base power usage. Thus, if the commanded speeds are s_l and s_r (which are in the range [0,1]), the power input is p, the time step is Δ_t, the base usage is k_{base} and the motor power factor is k_m, then the charge C_t and modulator h_t are given by

$$C_t = \text{clamp}\left(C_{t-1} + \Delta_t\big(k_{power}p - (k_{base} + k_m(s_l + s_r))\big)\right) \quad (2)$$

$$\text{where } \text{clamp}(x) = \max\big(0, \min(1, x)\big)$$

$$h_t = 1 - C_t \quad (3)$$

The power input p is obtained from the sum of the pixel inputs, multiplied by the charging efficiency constant k_{power}: one of the variables modified in the robot experiments. The values used were $k_m = 0.01, k_{base} = 0.005, k_{power} = 0.0025$. Once 10 runs have been obtained for each of the 10 networks generated by the 3 methods, they are evaluated according to three metrics, which are combined together by $\sum_i \log_2 \left(\frac{m_i}{r_i}\right)^2$, where m_i is the obtained value for each measure and r_i is a reference value. This provides a useful measure whereby +1 is "twice as good" and -1 is "half as good" [9, 12]. The three measures are: the mean of the standard deviation across time windows of the data (to provide a measure of variability across the whole time to avoid artificially high values when the robot only moves in one section of the run), the distance travelled multiplied by the distance from the origin (the light is at the origin, this shows exploration), and the total time survived before the charge dropped to zero. This metric requires three reference values – the obvious value for r_t is the maximum run time, the others were determined from the best runs.

$$m_d = \frac{\sum_{i=1}^{N} \sigma\left(d_{(i-1)s<t<is}\right)}{N}, \quad s = t_{max}/N, d_i = \sqrt{({x_i}^2 + {y_i}^2)} \quad (4)$$

(time-windowed standard deviation of distance)

$$m_t = \max t \quad (5)$$

(time survived)

$$m_T = \sum_{j}^{n-1} \left(\sqrt{(x_j - x_{j+1})^2 + (y_j - y_{j+1})^2} \sqrt{x_j^2 + y_j^2} \right) \tag{6}$$

(distance travelled biased for edge, for exploration)

$$S = \frac{1}{3} \left(\log_2 \left(\frac{m_d}{r_d} \right)^2 + \log_2 \left(\frac{m_t}{r_t} \right)^2 + \log_2 \left(\frac{m_T}{r_T} \right)^2 \right) \tag{7}$$

where N is the number of segments for windowing the SD and was set to 10 for these experiments, m_* indicates individual metrics, r_* indicates the reference values: $r_d = 2.45, r_t = 1000\text{s}, r_T = 2183$. The values for r_d and r_T are just over the maxima achieved across all the runs.

4.2 Simulator results

The metric and sub-metric results of all the simulator runs are shown in Fig. 1. According to the (somewhat arbitrary) combined metric, UESMANN and h-as-input perform better than output blending, but there is little difference between the mean performance of the former two. In the individual metrics, h-as-input has much more variation in performance, with several networks performing consistently very well while others often fail. UESMANN also shows variation, with fewer failing runs but fewer extremely good performers. Output blending is very consistent and always survives the full time, but doesn't appear to travel far.

Typical runs of the best performing networks for all three methods are shown in Fig. 3. Clearly the best performing network is h-as-input, and this can be seen in Fig. 1: the best network performs much better than UESMANN, but other poor networks pull the overall performance down. All networks rapidly converge to a limit cycle, as shown in the distance/charge phase plots. With output blending, this is a very tight figure-8 around the origin. This is because output blending is always performing some combination of phototaxis and wandering, and at anything but a very high charge phototaxis will always draw it back to the light. Both h-as-input and UESMANN appear to have a narrower transition region (as shown in previous experiments in [1]), and so show distinctive phase changes: when the charge drops below a certain point, the behaviour will change to light-seeking; and when the charge is high enough, the robot will wander again. The triangular shape in phase space is a consequence of this: the left-hand side of the triangle is phototaxis, the base is recharging stopped at the light, and the right-hand side is exploration. The double peak in h-as-input, and the initial irregularity, is due to the robot wandering far enough to "bounce" off the walls of the arena. UESMANN appears to show a much more "chaotic" limit cycle than h-as-input, which gives the positional plot a more "random" appearance: whereas h-as-input follows a fixed course, UESMANN varies more.

While UESMANN does not perform as well, it is considerably more conservative than h-as-input: in the latter, the transition to phototaxis occurs at $C \approx 0.3$ in h-as-input, leaving it at $C \approx 0.14$ at the return. In a larger arena h-as-input may run out of charge – here it has been helped by the "bounce" off the far wall.

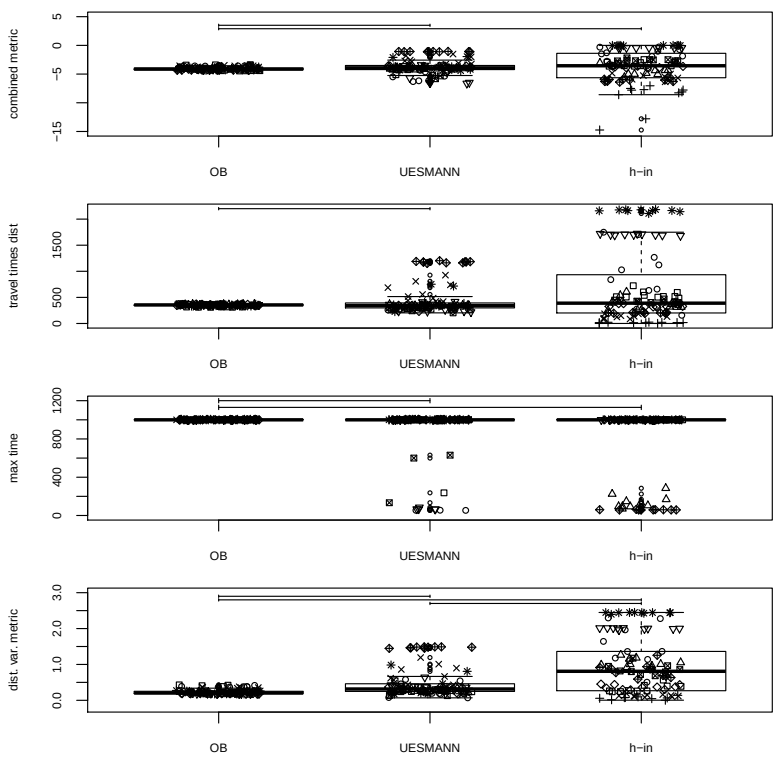


Fig. 1. Box plots of all runs of all networks, showing the combined and submetrics. The point shape indicates the network from which the run comes. Significant differences are indicated by brackets (Mann-Whitney U-test, $p < 0.05$).

UESMANN transitions to phototaxis at $C \approx 0.55$, leaving it with $C \approx 0.4$ at the return.

One interesting feature in the UESMANN run is the "blip" in the base of the triangle in the phase plot, which manifests in the position plot as a tiny inner loop: at a certain point in the recharge phase, the robot performs a "microexcursion", and then returns to recharge. This appears to be a consequence of the interesting dynamics of the UESMANN architecture, and was present in all runs of this particular network.

In order to examine the transtional behaviour, the three best networks by combined metric were each run for 100 seconds, for 100 different fixed modulator values 0 to 1. This was repeated 50 times, and the mean distance-from-light of the all the runs at each modulator level plotted. No charge model was used. If the network is predominantly wandering at that modulator level, the mean distance should be high; if performing phototaxis, the mean distance should be low. The results are shown in Fig.2. Output blending shows the expected gradual shift from long distances to short, while h-as-input shows a transition of

width around 0.2. UESMANN is interesting: while it follows the trend of output blending (the "ideal" blend between two networks), it shows a very complex transitional behaviour with the robot changing priorities between phototaxis and wandering in distinct stages, a pattern we came to call "dithering". This pattern proved useful in the real robot by supplying a source of complexity allowing the robot to follow different courses in different runs and by helping with a steering problem, as the next section will show.

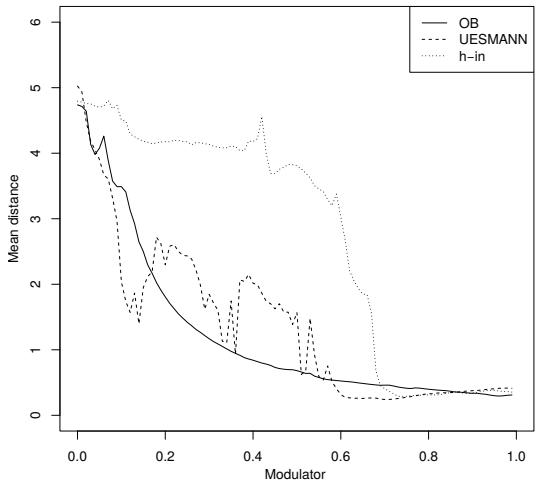
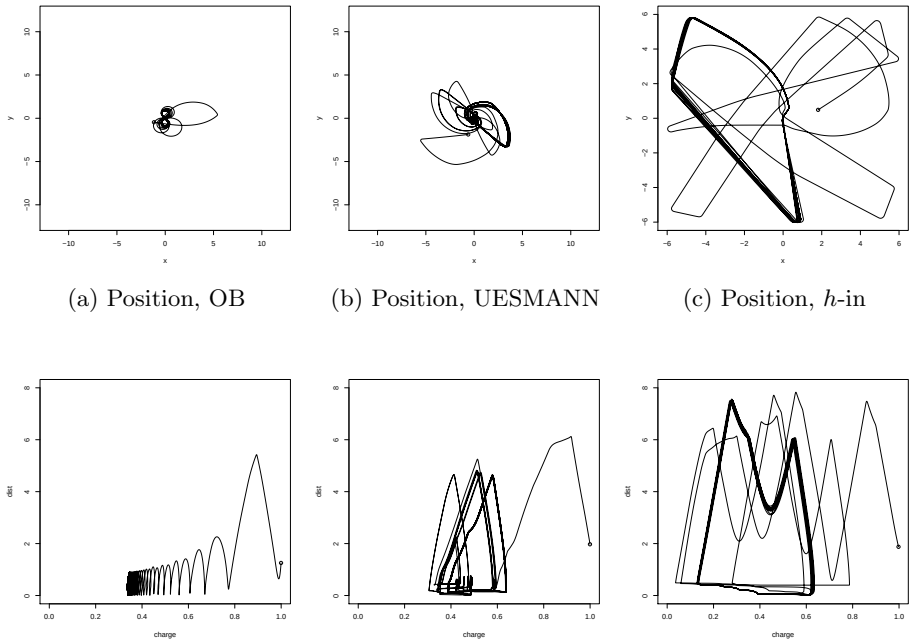


Fig. 2. Mean distance at different hormone (i.e. modulator) levels for 50 runs of the simulated robot.

5 Robot experiments

The best networks (according to the combined metric) for the three network types were taken forward into the robot experiments. The robot used was a Pioneer 2DX: a differential-drive robot as in the simulator, with 2 large driven wheels and a caster wheel at the back. It was fitted with an omnidirectional camera to act as a light sensor, and carried its own sonar sensors of which the front 8 were used, which were oriented the same way as in the simulator. The network software was run on a host laptop, communicating over TCP to the robot, which appeared as a ROS node on the laptop.

The light source was a 60W incandescent bulb in an opaque hood, suspended $\sim 1.5m$ above the floor, shining a diffuse circle on the floor. The on-board omnidirectional camera image was gaussian blurred and summed across 32 radii,



(a) Position, OB (b) Position, UESMANN (c) Position, h-in

(d) dist/charge, OB (e) dist/charge, UESMANN (f) dist/charge, h-in

Fig. 3. Typical runs of best seeds in simulation, position and distance from light/charge

giving 32 pixels around the robot. This was downsampled to 8 pixels on the laptop. Fig. 4 shows a picture of the robot with the light source and two on-board camera views, one for far and one for near light.

(a) Robot and light source (b) Camera image, blurred, far light (c) Camera image, blurred, near light

Fig. 4. Robot and light source, and two omnidirectional camera views showing dots for the 32 summed radii.

Positional tracking was done by mounting a diffused red LED on the robot (above the centre of rotation) and using a commodity webcam, filtering for red blobs, finding the centroid of the largest blob and transforming from screen space to a plane in 3D space by perspective transform. The accuracy of the tracking was from $\sim$ 0.01m close to the camera, to $\sim$ 0.2m at the far distance. The tracking data was captured at a frequency of roughly 1Hz, and sent over ROS to the main program for logging purposes. Some parts of the arena were not visible to tracking – notably a long section in which the LED was obscured by the light source hood. In the logs, points for which no tracking data were received have the same position as the last logged point.

The arena was bounded by a (nearly) sonar-opaque mesh, and was of an irregular, roughly triangular shape providing the largest possible area. The light source was suspended (due to environmental constraints) over the narrow end, and is indicated by a circle in the figures below. Thus, the experiments often involve cycling around the "safe" narrow area with occasional excursions into the wide, dark area.

12 experiments were done, with five repetitions each. The factors were: network type (OB/UESMANN/h-as-input), initial direction ("south" into the narrow end or "north" into the dark area) and two values of k_{power}: the constant used to determine how efficiently the simulated battery charges. The values chosen were 0.0025 and 0.003. Otherwise the same power model was used. One major difference is that the speeds were scaled down: 1ms^{-1} is not a viable speed for a Pioneer. The outputs of the network s_l, s_r were multiplied by 0.05. Naturally this makes it difficult to compare the results with the simulator, any comparisons made must be qualitative.

5.1 Results

During the runs for h-as-input it was found that the best network in simulation performed extremely badly, colliding frequently with the mesh and putting the robot and environment in danger. Only eight runs (two for each k_{power} and direction) were performed before the network was abandoned. Three of these runs, showing typical behaviour, are shown in Fig. 5. Note that the arena boundary is approximate: in all these runs, the robot became entangled in the mesh. It seems that this particular network responded very late to sonar, which allowed it to traverse very far in simulation but caused problems with the very noisy sonar on the robot (see below). Therefore the second-best network was used for all remaining experiments, which was rather more conservative.

Below we present a largely descriptive analysis of the behaviours of the different networks. The survival times for all runs are shown in Table 1, and the performance metric used in the simulation experiments (with new reference values obtained from the raw results) is shown in Table 2. Both tables are sorted by the means. Note that the ordering is nearly the same: h-as-input is penalised by the metric because it does not travel far. The other sub-metrics show very similar ordering. The mean metrics for all runs are shown in Fig. 3.

Table 1. Survival times for all runs, sorted by mean. The number in the row name refers to k_{power}.

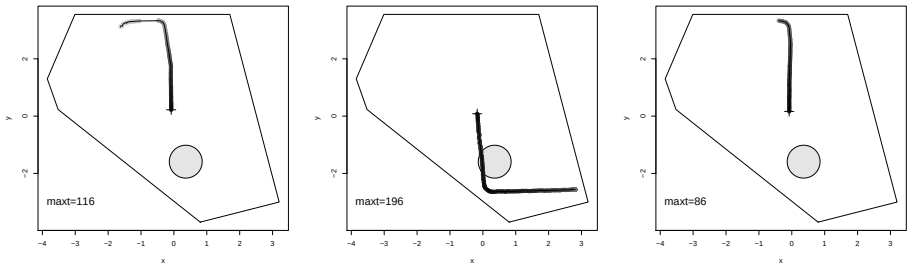
name	r1	r2	r3	r4	r5	mean
h-in south 0.0025	1000	1000	1000	1000	1000	1000.00
OB north 0.003	946	1000	1000	1000	1000	989.28
UESMANN south 0.0025	1000	1000	636	1000	1000	927.16
UESMANN south 0.003	1000	1000	1000	955	521	895.30
OB south 0.0025	649	1000	596	639	619	700.64
OB south 0.003	167	383	392	1000	780	544.30
h-in south 0.003	361	364	398	399	884	481.16
UESMANN north 0.003	1000	176	183	169	170	339.62
h-in north 0.003	147	167	150	157	149	153.82
OB north 0.0025	126	122	122	125	120	123.10
UESMANN north 0.0025	117	118	115	116	118	116.78
h-in north 0.0025	110	113	114	112	110	111.74

Table 2. Combined performance metric for all runs, sorted by mean.

name	r1	r2	r3	r4	r5	mean
OB north 0.003	-0.15	-0.21	-0.45	-0.40	-0.34	-0.31
h-in south 0.0025	-0.72	-0.66	-0.23	-0.48	-0.35	-0.49
UESMANN south 0.0025	-0.39	-0.48	-1.77	-0.60	-0.65	-0.78
UESMANN south 0.003	-0.31	-0.52	-1.13	-1.16	-2.47	-1.12
OB south 0.0025	-1.69	-0.67	-1.64	-2.04	-1.67	-1.54
OB south 0.003	-5.77	-2.73	-2.68	-0.46	-1.05	-2.53
h-in south 0.003	-2.86	-3.16	-3.02	-3.08	-1.30	-2.68
UESMANN north 0.003	-0.28	-4.92	-5.11	-5.11	-5.12	-4.11
h-in north 0.003	-4.31	-5.53	-5.84	-5.86	-5.91	-5.49
OB north 0.0025	-5.85	-6.15	-5.86	-5.87	-6.01	-5.95
UESMANN north 0.0025	-5.85	-5.90	-5.86	-5.99	-6.27	-5.97
h-in north 0.0025	-6.70	-6.69	-6.64	-6.57	-6.66	-6.65

Table 3. Means of all metrics for all runs.

	survival time	windowed SD	edge-biased travel	combined metric
OB	589.33	0.42	79.97	-2.58
UESMANN	569.71	0.36	73.03	-2.99
h-as-input	436.68	0.29	57.03	-3.83



(a) $k_{power} = 0.0025$, north (b) $k_{power} = 0.0025$, south (c) $k_{power} = 0.003$, north

Fig. 5. h-as-input runs from the best simulator network. The arena walls are shown as lines in the position plot, and the light source by a grey circle. Charge is indicated by shading (the paler, the lower). The maximum achieved time is also shown.

Looking at the metrics can be deceptive, because survival and coverage is often dependent on serendipitous sonar bounces and the modulator level at which transitions occur, which can easily be changed by modifying the modulator in some way. We are more interested in the nature of the transition between behaviours and how it helps or hinders in this application.

However, there are some general observations to be made. Overall, output blending fares only slightly better than UESMANN while h-as-input is worst. This is surprising, given that UESMANN is using a single network with the same number of weights as each of the two networks used in OB, and has fewer weights than the h-as-input network (although we are using the second-best h-in network).

No network does well starting into the dark ("north") with low power: they all turn too late to return in time to recharge, but the nature of these turns is quite different. UESMANN does well facing initially into the light, while output blending does not. Facing into the dark, the position is reversed: output blending has a tendency to always follow the same track, which leads it into trouble facing south: it finds a path into the dark area and cannot return. UESMANN follows a variety of paths and is more conservative.

All networks occasionally touch the mesh, but turn out of it fairly quickly (unlike the original h-as-input network). Tests show that this may be due to the nature of the mesh and the walls behind it, which are strongly sonar-reflective: the sonars sometimes get echoes from the wall instead of the mesh.

Output blending Fig. 6 shows the position and phase plots for typical output blending network runs. We are no longer seeing the tight cycling which this network performed in simulation: this is because the differential steering is not perfect in the robot. Tests show that the robot will continue to drive straight while one motor's requested speed is reduced, until the speeds differ by around 0.3 (if the other is at 1). Thus, the robot can make straight runs even when

it is being commanded (partially) towards the light. The environment here is imposing a narrower transition between the behaviours than is being produced by the network outputs.

In general, the runs are very similar within each experiment, although one run south at $k_{power} = 0.003$ succeeds where the others fail by fortuitously being turned towards the light slightly earlier by sonar reflection.

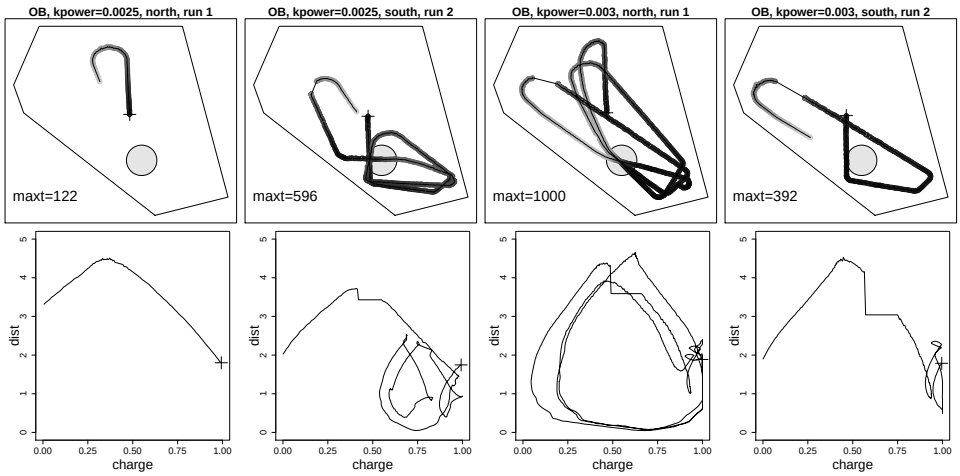


Fig. 6. Typical runs for output blending, position and phase plots. See Fig. 5 for an explanation.

UESMANN Fig. 7 shows the results for UESMANN runs. The turns are tighter here – inspection of the variable data shows that the motor responses are varying more during transitions, showing the "dithering" behaviour described earlier in simulation. These slight oscillations between the two behaviours through the transition seem to counteract the steering lag which affects output blending by providing higher rotational accelerations in the motor. UESMANN's run also appear more "chaotic" (in the informal sense), with the robot following different routes in each run by turning at slightly different points and angles. This is also likely to be due to the complex transition behaviour – small differences in the modulator can lead to larger changes in behaviour than in the other networks, although the overall transition is still gradual.

This is particularly notable in south-facing runs with $k_{power} = 0.003$, a particularly interesting instance of which is shown. Here, the robot made its way to the far corner, stopped to recharge (there is enough light to do so if the motors are not turning), and then returned. Stopping in darkness seems to be an emergent property of the network (note that the robot did not actually hit the mesh

in this instance – the left-top corner of the arena is slightly further to the left in reality).

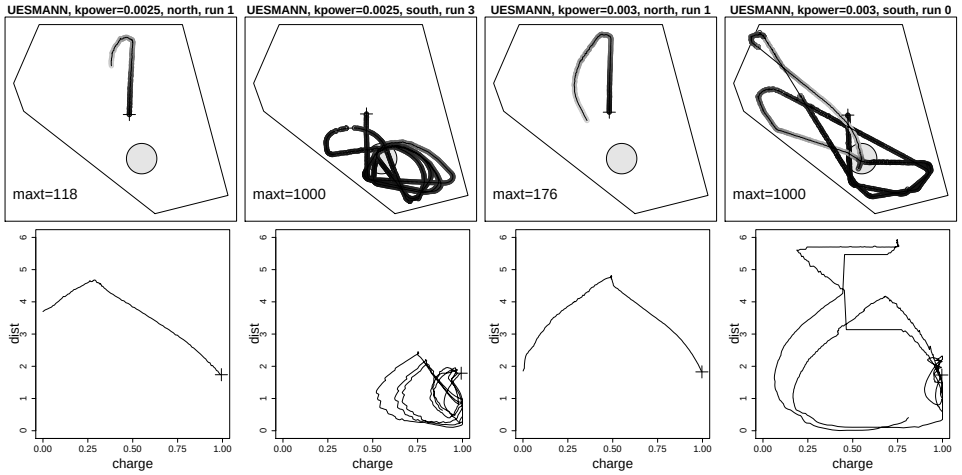


Fig. 7. Typical runs for UESMANN, position and phase plots. The arena walls are shown as lines in the position plot, and the light source by a grey circle. Charge is indicated by shading (the paler, the lower). The maximum achieved time is also shown.

***h*-as-input** Fig. 8 shows the results for this network. This is a conservative network, as can be seen from the low power 0.0025 plots. All the runs are very similar within each experiment. At high power, some problems can be seen. The nature of the turn running north is unusual, with a partial sonar turn followed by a turn towards the light. The slight drop in power may potentiate the network towards taking action on a sonar signal here. Many of the south-facing high power runs resulted in running into the mesh, as shown here. This is odd, because both behaviours should drive it to turn in the same direction at this point. It should be borne in mind, however, that this is the second-best h-as-input network – although the first also had difficulties with turning (see the opening of this section).

Analysis of motor differential To investigate the behaviour of the motors, a histogram of $|s_l - s_r|$ was plotted for all runs: see Fig.9. Output blending shows large variation in the differential, with many intermediate values used as the subnetworks blend their outputs, but heavily weighted towards smaller values. UESMANN shows a similar effect, but the distribution is much more even across the range (apart from 0 and 1). The h-as-input network shows a tendency to go either straight ahead (0) or turn (1). UESMANN's higher use of the entire range of motor differentials may reflect increased complexity in its behaviour.

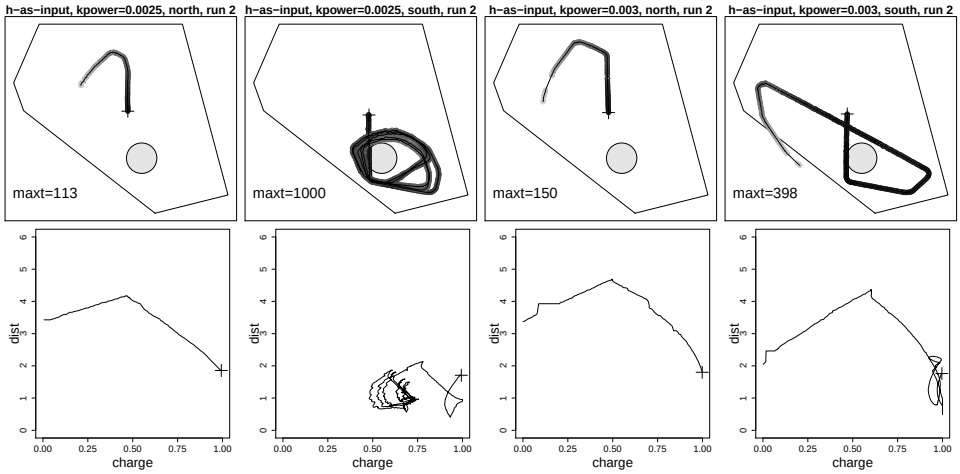


Fig. 8. Typical runs for UESMANN, position and phase plots. The arena walls are shown as lines in the position plot, and the light source by a grey circle. Charge is indicated by shading (the paler, the lower). The maximum achieved time is also shown.

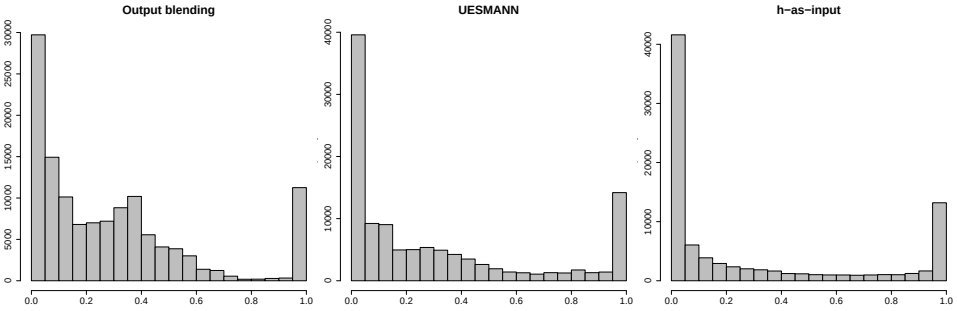


Fig. 9. Histogram of motor differential across all runs

6 Conclusion

It is often useful to move smoothly from one behaviour to another, particularly in embodied systems. While it is generally true that a small change in environment should lead to a small response in behaviour, it is also true that some complexity in the transition between behaviours can be useful. It may allow the system to find fortuitious escapes from difficult situations, or new ways of exploring and exploiting its environment.

Whereas naïve output blending produces a smooth, "ideal" transition, and using the modulator as just another input gives a fairly sharp transition (and sometimes does not learn well) the UESMANN architecture seems to provide

such a complex transition, while maintaining the essential nature of the blend between the behaviours.

It is remarkable that UESMANN is able to learn two complex functions fairly well using the same number of weights as a network which performs one of those functions only a little better, and with such a simple modulatory and learning technique. Currently work is being done on analysing the behaviour of single UESMANN nodes to gain insights into how this is achieved, starting at first principles with 2-2-1 networks blending boolean operators. The study of such a simple system's capacity to learn multiple behaviours may yield important results for more complex systems. Future work should include testing the current application with lower hidden node counts, before moving onto dynamical systems analysis of the UESMANN training process and the resulting networks, consideration of other learning methods (such as artificial evolution) and incorporating UESMANN-like layers into deep learning networks to perform multiple functions.

References

1. Finnis, J.C., Neal, M.: UESMANN: A feed-forward network capable of learning multiple functions. In: International Conference on Simulation of Adaptive Behavior. 101–112. Springer (2016)
2. Funahashi, K., Nakamura, Y.: Approximation of dynamical systems by continuous time recurrent neural networks. Neural networks 6(6), 801–806 (1993)
3. Husbands, P., Philippides, A., Smith, T., O'Shea, M.: Volume signalling in real and robot nervous systems. Theory in Biosciences 120(3-4), 253–269 (Dec 2001)
4. Kaczmarek, L.K., Levitan, I.B.: Neuromodulation: the biochemical control of neuronal excitability. Oxford University Press, New York (1987)
5. Magg, S., Philippides, A.: Gasnets and CTRNNs–a comparison in terms of evolvability. In: From Animals to Animats 9, 461–472. Springer (2006)
6. Moioli, R.C., Vargas, P.A., Von Zuben, F.J., Husbands, P.: Towards the evolution of an artificial homeostatic system. In: IEEE Congress on Evolutionary Computation. 4023–4030. IEEE (2008)
7. Neal, M.: Once more unto the breach: Towards artificial homeostasis. In: De Castro, L.N., Von Zuben, F.J. (eds.) Recent Developments in Biologically Inspired Computing, 340–365. Idea Group (2005)
8. Neal, M., Timmis, J.: Timidity: a useful emotional mechanism for robot control? Informatica (Slovenia) 27(2), 197–204 (2003)
9. Rodriguez, G., Weisbin, C.R.: A new method to evaluate human-robot system performance. Autonomous Robots 14(2-3), 165–178 (2003)
10. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature 323(6088), 533–536 (1986)
11. Sauze, C., Neal, M.: Artificial endocrine controller for power management in robotic systems. Neural Networks and Learning Systems, IEEE Transactions on 24(12), 1973–1985 (2013)
12. Tunstel, E.: Operational performance metrics for Mars exploration rovers. Journal of Field Robotics 24(8-9), 651–670 (2007)
13. Vargas, P.A., Di Paolo, E.A., Husbands, P.: Preliminary investigations on the evolvability of a non spatial gasnet model. In: Advances in Artificial Life, 966–975. Springer (2007)