

First Steps in Evolving Path Integration in Simulation

Robert Vickerstaff

Centre for Computational Neuroscience and Robotics, School of Biological Sciences,
University of Sussex, Brighton, BN1 9QG, UK

Abstract. Path integration is a widely used method of navigation in nature whereby an animal continuously tracks its location by integrating its motion over the course of a journey. Many mathematical models of this process exist, as do at least two hand designed neural network models. Two one dimensional distance measuring tasks are here presented as a simplified analogy of path integration and as a first step towards producing a neuron-based model of full path integration constructed entirely by artificial evolution. Simulated agents are evolved capable of measuring the distance they have travelled along a one dimensional space. The resulting neural mechanisms are analysed and discussed, along with the prospects of producing a full model using the same methodology.

1 Introduction

Path integration (PI) is a class of navigation strategy whereby an animal tracks its location by integrating its velocity (or the equivalent such as double integrating its acceleration, counting steps taken etc.) over the course of a journey. It is known to be used by many animal species, being particularly well studied in insects (see e.g. [1]). For some species it may be the only available method of navigation, as seems to be the case for the desert ant *Cataglyphis fortis* (see e.g. [2]), for others it is used in addition to or closely coupled with other methods such as land mark recognition [3]. A series of mathematical models of PI have been produced describing different solutions to the problem, usually based on either polar or Cartesian two dimensional coordinate systems. Two hand designed neural network models have also provided existence proofs of a system based on neuron-like elements.

In this paper we describe work aimed at producing another neuron-based PI model, but one constructed automatically using a genetic algorithm (GA) - to our knowledge an approach that has not been employed for modelling this problem before. It is hoped that a PI system produced by an evolutionary search method will complement the existing models, in that, while the resulting network may be harder to analyse, it will not be so directly influenced by the biases of the designer. In fact an evolved PI network may be more simple than a hand designed one, since a wider range of network styles and architectures can be considered using a GA than by hand design, making a wider range of neural mechanisms available for the solution.

Presented here are the results of a first step towards such a PI system, based on two simple distance measuring problems (used as a one dimensional analogy of PI in higher dimensions). Existing PI models are presented in section 2; the tasks and neural networks used for the present work are described in section 3; the evolved neural networks are described and tested in section 4 and the prospects for modelling PI in two dimensions using the same approach are discussed in section 5.

2 Existing Path Integration Models

Most existing PI models are mathematical descriptions of the calculations required to update a two dimensional position vector of the nest's location relative to the agent or vice versa. The equations are usually iterative formulae describing the vector's components at time $t + 1$ given their values at time t and the agent's motion between t and $t + 1$ written using either polar or Cartesian coordinate systems. The animal is assumed to be moving on a flat two dimensional plain. For reviews see [5, 4].

At least two neuron based models also exist ([7, 6]), whereby the calculations are carried out using networks of simple model neurons. Wittmann and Schwelger [7] solve the problem using a nest centred polar coordinate system which performs accurate PI. The vector is elegantly encoded using a sinusoidal array - an array of neurons whose activity pattern describes a sine wave in space. The wave's amplitude encodes distance from the nest, while the phase encodes bearing. Updating the vector is achieved by simple element-to-element addition of two such arrays. Hartmann and Wehner [6] also solve the problem using nest centred polar coordinates, but their system uses an approximate method designed to mimic the systematic errors seen in ant PI. The vector is represented separately as a distance and a bearing: the distance is encoded as an activity pattern extending and receding along a chain of neurons, while the bearing is encoded as a patch of activity moving clockwise and anti-clockwise around a circle of neurons.

The work presented here is not yet at a stage where direct comparison with either of the above classes of model is possible, for example the issue of what type of coordinate system to use does not arise since only a scalar value is needed. However, the method used to encode this scalar value can be compared to the two neural models. Is the neuron firing rate used or some other property of the dynamics? Is the value encoded using a single neuron or several? Also, the effect of modelling sensing errors or noise within the neural network can be studied as was the case with at least one of the mathematical models (Benhamou *et al.* (1990) cited in [5]).

3 Methods

A continuous time recurrent neural network (CTRNN, [8]) was used to control the agent. The neural network was defined by a genotype under the control of a

genetic algorithm. The network received input from a speed sensor and a food sensor (which signaled when the agent had reached its goal and should return to the nest), and controlled three actuators: a forward motor, a reverse motor and a signal (which the network must activate when the agent has returned to the nest). The agent moved as a particle in a one dimensional space controlled by simple physical modelling. Key characteristics of each agent’s behaviour were recorded during trials and used to assign a fitness to the neural controller at the end of each trial. Two tasks were set for the agent in separate experiments: (i) the two-way task: move forwards until reaching a randomly placed food item (indicated by the food sensor input changing from 0 to 1), return to the starting point and signal (defined as the firing rate of the signal neuron changing from a supra-threshold to a super-threshold level); (ii) the one-way task: move forward until reaching a randomly placed food item, then continue past the item for a distance equal to that already travelled before signalling. In both cases the location the agent is required to signal at is referred to as the nest for simplicity.

The CTRNNs used model a neural network using a continuous time description of neural firing rates, using the differential equation (adapted from [8]):

$$\tau_i \frac{dv_i}{dt} = -v_i + \sum_{j=1}^{N_{inputs}} w_{ji} \sigma_j \quad (1)$$

where $i \in \{1, \dots, N_{neurons}\}$, τ_i is a time constant, v_i is the cell membrane potential, w_{ji} is the weighting for input j of neuron i ($N_{inputs} = 10$ for all results presented here) and σ_j is the current value of input j . The input may be from a neuron or sensor. For a neuron $\sigma_j = \frac{1}{1 + e^{-(g_k(v_k + b_k))}}$ where k is the inputting neuron and g and b are gain and bias parameters respectively. For a sensor the input is the current sensor activation (i.e. the speed or food sensor) linearly mapped such that $(\sigma_j = I_k) \in [0, 1]$ under all possible conditions, where k is the inputting sensor.

τ_i , w_{ji} , g_i , b_i and the type (neuron or sensor) and subscript (k) of the input are all evolvable properties controlled by the genotype. τ_i and g_i are encoded in \log_{10} form and raised to the power of ten before being used. Cell potentials are initialised at the beginning of each simulation run to values, v_{0i} , also encoded by the agent’s genotype. A neuron is assigned to control each of the three outputs (forward motor, reverse motor, signal), control can be switched to other neurons by mutation.

Three global parameters control the amount of noise in the simulation. η_v controls uniform noise in the initialisation of the cell potentials such that $v_i \in [v_{0i} - \eta_v, v_{0i} + \eta_v]$. η_I controls uniform sensory noise whereby $I'_k \in [I_k - \eta_I, I_k + \eta_I]$ each time step. η_σ controls uniform neural noise whereby $\sigma'_k \in [\sigma_k - \eta_\sigma, \sigma_k + \eta_\sigma]$ each time step. The equations were numerically integrated using Euler’s method with a step size of 0.2 while the genetic algorithm was running, evolved agents were tested with a step size of 0.2 and 0.01 to check for artifacts due to numerical integration (all subsequent analysis was done with step size 0.2 since the level of noise is dependent on the step size).

The agent’s location in the one dimensional arena is specified as a single point with zero volume, x . It’s motion is influenced by the forces exerted by its two motors and by *wind* which can exert a drag force on the agent. The wind speed is a further source of random noise in the simulation, being set to a uniform random value from the interval $[-1, 1]$ for a uniform random length of time $\in [0.2, 20]$ time units (not integration steps). Wind speeds are positive in the same direction as positive agent speeds. The agent’s motion is controlled by Newtonian dynamics assuming a linear drag model:

$$F = F_{max}(\sigma_f - \sigma_r) - C_{drag}\left(\frac{dx}{dt} - w\right), \quad \frac{d^2x}{dt^2} = F/(M_{agent} + M_{food})$$

where F is the resultant force, F_{max} is the maximum force a motor can exert (set to 1), σ_f, σ_r are the firing rates of the forward and reverse motor control neurons respectively, C_{drag} is the linear drag coefficient (set to 1), $\frac{dx}{dt}$ is the agent’s speed, w is the wind speed, $\frac{d^2x}{dt^2}$ is the agent’s acceleration, M_{agent} is the agent’s mass (set to 1) and M_{food} is the mass of food carried by the agent, equal to 0 before it has reached to food and 1 after that. This equation was integrated using Euler’s method to give the speed, then integrated again exactly (i.e. assuming only a constant acceleration) to give the agent’s new location.

A generational genetic algorithm (GA), [9], population size 15, was used to evolve the agent’s neural network. The fittest 5 genotypes were copied, mutated and used to replace the worst 5 genotypes each generation. Each agent was evaluated using 5 trials and assigned the fitness of the worst trial.

Fitness was calculated as a value $\in [0, 1]$ using a series of formulae reflecting key data collected during the trial: nearest approach to food, furthest distance from food, time reached food, nearest approach to nest, furthest distance from nest, time reached nest, minimum value of signal neuron, maximum value of signal neuron, trial status when signalled, location when signalled. The fitness rewards were calculated for three criteria: (i)if and when it reached the food, (ii)if and when it reached the nest (after first reaching the food) and (iii)if and when it signalled the nest location. For (i) high values for nearest-to-food and furthest-from-food were penalised, high values for time-to-food were penalised but always gave a higher reward than not reaching the food at all. Likewise for nearest-to-nest and furthest-from-nest in (ii), likewise reaching the nest was always better than having only reached the food. For (iii) if the agent didn’t signal rewards were given for higher maximum and lower minimum values of the signal neuron, but signalling anywhere was always rewarded more than not signalling at all; signalling after reaching the food was better than signalling before it. All rewards were calculated such that they were scaled to a well defined interval, and scaled relative to each other so that 94% of the maximum fitness (1.0) was accounted for by the accuracy of signalling the nest location after first reaching the food, the other criteria being present to encourage the earlier stages of behaviour to evolve first. Signalling accuracy was assessed by an absolute rather than relative error measure, rewards were therefore normalised relative to the size of the food placement zone rather than the actual distance between the

food and nest in a given trial. The location of the food for each of the five trials was set using a stratified random scheme, such that the defined food placement zone was divided into five equal sections, each being sampled randomly in one of the trials. Agent’s were initially evolved to perform the task with the food zone set between 5 and 10 distance units. Once satisfactory results were obtained the food zone was doubled in length successively until a zone from 5 to 400 distance units was reached.

The GA used was capable of varying the number of neurons in the network using neuron duplication and deletion operators; this feature was used in the experiments where the network size was allowed to vary. Other neuron level mutation operators were used in all experiments: homologous recombination (replacement by the most similar neuron in another genotype), overwriting (deletion followed by duplication of a remaining neuron, to allow duplication in a fixed length genotype) and randomisation (deletion followed by replacement by a randomly generated neuron). Neuron parameter mutations were performed using creep mutations for floating point values, where the new value was the old value plus a Gaussian random value with zero mean whose standard deviation was scaled by the size of the valid range of the parameter and a global mutation rate parameter.

4 Results and Analysis

Initially two sets of experiments were performed for each of the two tasks, for both of which neural network size was fixed to 3 (one to control each of the actuators). These were: (i) with wind, but no other sources of noise ($\eta_v = 0, \eta_I = 0, \eta_\sigma = 0$); (ii) as (i) but $\eta_v = 1, \eta_I = 0.25, \eta_\sigma = 0.025$. GA runs were also performed where the number of neurons was allowed to change by mutation, but for both tasks these runs did not produce any significant improvement in agent fitness or novel mechanisms compared to the agents with only three neurons.

The Two-way Task. Here the agent must move forward to the food, then return to the starting point and signal. In the experiments without noise agents evolved able to signal the nest location accurately, in spite of the buffeting effects of the wind. Five duplicate GA runs were performed, all of which evolved the same method of solving the task. For the best solutions the fitness was greater than 0.99 no matter where the food was within the placement zone, with the worst ones still consistently greater then 0.95. The forward motor neuron flipped from high to low when the food was reached, likewise the reverse motor neuron flipped from low to high, bringing about the agent’s change of direction. This can be achieved by a simple reactive mechanism since the food sensor input remains low until the food is reached, then becomes high until the end of the trial¹. The signal neuron acted as an integrator of the speed sensor input. This is possible

¹ Earlier runs (data not shown) had been tried where the food sensor only stayed high within a short distance of the food; agents evolved successfully to perform under this scheme; it was changed to the current scheme to simplify subsequent analysis slightly

for a single neuron since, if the $-v_i$ term in equation (1) is small, then the neuron simply integrates the sum of its weighted inputs. The best agents used a very small range of v to encode the agent's location, to minimise the decay and therefore perform more accurate integration. This came at the price of being very sensitive to noise if introduced, especially where $\eta_v > 0$, since this acted to effectively randomise the neuron's state at the start of the trial. For example setting $\eta_v = 0.1$ resulted in fitnesses varying between 0.05 and 1.0 for trips up to 150 distance units and between 0.6 and 1.0 for trips between 150 and 400 units. The best agents were also able to flip the two motor states rapidly once the food was reached. The less efficient solutions entered a short phase where only one of the motor controllers had flipped states, resulting in no net force being produced by the motors. For the best solutions, the agent's location varied linearly with the integrator neuron's state, see Fig. (1, left) for the network of the fittest agent.

To test the generality of the integration mechanism, trials were performed where the agent was held still once it had reached the food for 500 time units before being allowed to return as normal. This reduced the fitness of the agent only marginally to around 0.9 for short journeys, and had even less effect on longer runs.

The experiments with noise produced similar results, except that the signal neuron encoded the agent's location using a cell potential range approximately 50 times larger, (approximately 3.3 v units per 100 distance units compared to approximately 0.065 per 100). This reduced the effects of initial neuron potential noise. Agents scored greater than 0.9 for most trials, dropping to 0.85 for the longest trips. Plots of agent location against signal potential were still noticeably linear, but included substantial wobbles (data not shown). For short journeys the agents were prone to signal very early before reaching the food due to noise pushing the signal neuron above threshold.

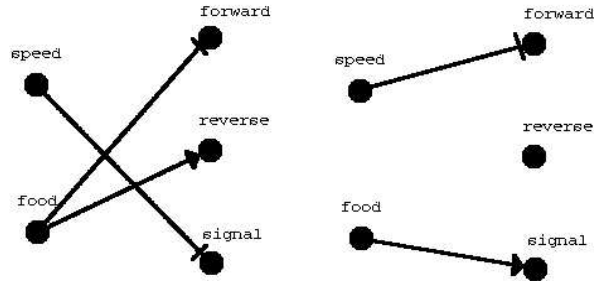


Fig. 1. Evolved networks which solve (left) the two-way and (right) the one-way tasks. All redundant connections have been pruned. Excitatory connections are shown with arrowheads, inhibitory ones with bars. Sensors are on the left of each network, actuator control neurons are on the right.

The One-way Task. This task involved simply travelling forward from the starting point, passing the food and signalling when an equal distance past the food had been travelled. This makes the simple one neuron integrator solution impossible since the two legs of the journey do not result in different speed sensor inputs. The experiment without noise evolved the same solution in all five runs. It varied its forward motor output in a way which damped down the effects of the wind on the agent's speed whereby the motor output dropped rapidly once the speed sensor got above approximately 0.8, and rose once it was below approximately 0.55. The signal neuron did not receive input from the speed sensor, only from the food sensor. Initially the signal neuron's v decreased very slowly due to a large τ and low negative $\frac{dv}{dt}$, at an approximately linear rate. Upon reaching the food the food sensor input became high, and the v increased at an approximately linear rate due to this change, see Fig.(1, right) for the network of the fittest agent. The agent was therefore only measuring time and not integrating the speed sensor input. By reducing speed variations with compensatory motor output this process approximates integration of the speed. The accuracy of this agent is approximately the same as the three neuron two-way task agent evolved without noise, with fitnesses of greater than 0.98 regardless of the location of the food. This mechanism simply 'assumes' the first and second legs of the journey will take the same amount of time to complete. In trials where the wind speed was held at 0.5 for the first leg and at -0.5 for the second leg the agent consistently signalled too early.

The experiment with noise did not evolve the wind damping strategy seen in the above experiment, but otherwise used the same time measuring mechanism. Fitness was markedly reduced, scoring between 1.0 and 0.85, except for short journeys where the signal was once again prone to be triggered by noise very early in the trial, resulting in very low fitnesses.

5 Discussion

The two distance measuring tasks were each solved by a different strategy. Of these the solution to the first task solves the problem in the most general way, in that it can truly integrate the speed sensor input. This relies on using a single neuron as an integrator, which is not entirely without precedence in the neuroscience literature [10]. Alternatively, each CTRNN neuron could be assumed to model a group of similar neurons. This result is markedly different from and more simple than the use of a chain of neurons to encode a scalar value [6], and comes quite naturally once neurons are described using a continuous time non-reactive model. This simple mechanism could be extended to perform PI in two dimensions by using two integrator neurons to record the agent's location as, for example, a nest-centred Cartesian coordinate, although it remains to be seen if such a mechanism could be produced by a GA. The agents performing the one-way task did not evolve a mechanism capable of generalised integration, but instead simplified the problem by using a negative feedback loop to travel at a more constant speed. This could of course have been modelled more abstractly

without concern for the dynamics of the agent's motion, by simply assuming a constant speed. In nature, however, an animal's motion and PI mechanism have evolved together. When testing the hypothesis that an animal's nervous system might only need to be capable of performing approximate PI to navigate successfully, explicit modelling of the agent's motion may prove beneficial for this reason. These results have shown two simple strategies for performing a simple analogy of path integration. The methods used here will now act as the basis for further work aimed at producing a two dimensional path integration model by evolutionary search.

References

1. Collett, M., Collett, T.S.: How do insects use path integration for their navigation? *Biol. Cybern.* **83** (2000) 245-259
2. Wehner, R., Gallizzi, K., Frei, C., Vesely, M.: Calibration processes in desert ant navigation: vector courses and systematic search. *J. Comp. Physiol. A* **188** (2002) 683-693
3. Wehner, R., Michel, B., Antonsen, P.: Visual navigation in insects: coupling of egocentric and geocentric information. *The Journal of Experimental Biology* **199** (1996) 129-140
4. Benhamou, S., Seguinot, V.: How to Find one's Way in the Labyrinth of Path Integration Models. *J. theor. Biol.* **174** (1995) 463-466
5. Maurer, R., Seguinot, V.: What is Modelling For? A Critical Review of the Models of Path Integration. *J. theor. Biol.* **175** (1995) 457-475
6. Hartmann, G., Wehner, R.: The ant's path integration system: a neural architecture. *Biol. Cybern.* **73** (1995) 482-497
7. Wittmann, T., Schwegler, H.: Path integration - a network model. *Biol. Cybern.* **73** (1995) 569-575
8. Beer, R.D.: Toward the evolution of dynamical neural networks for minimally cognitive behavior. In: Maes, P., Mataric, M., Meyer, J., Pollack, J., Wilson, S.(eds.): *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior*. MIT Press (1996) 421-429
9. Goldberg, D.E.: *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, Reading, Mass. (1989)
10. Egrov, A.V., Hamam, B.N., Fransen, E., Hassel, M.E., Alonso, A.A.: Graded persistent activity in entorhinal cortex neurons. *Nature* **420** (2002) 173-178