

A Neural Network Approach to Multi-vehicle Navigation

Geoffrey Fox, Eitan Gurewitz*, Yiu-fai Wong

Caltech Concurrent Computation Program
California Institute of Technology
Mail Code 206-49, Pasadena, CA 91125

Abstract

We develop a neural network formulation for multi-vehicle navigation on a two-dimensional surface. here. A time-linking map is generated for each individual vehicle using techniques similar to the known shortest path algorithms for an isolated vehicle. Neural networks are then applied to generate non-conflicting paths minimizing the time of travel.

1. Introduction

This paper presents a neural network approach to the multi-vehicle navigation problem. Here we use the term vehicle to refer to a point which travels on a surface of navigation (NS). Navigation as presented here refers to the determination of a path in the space-cost (time) domain from an origin to a destination point. The surface of the navigation usually has a terrain with position dependent velocities and/or hazards which the vehicle has to consider. The navigator searches for an optimal path on this surface. Optimum here may be with regard to minimal length, minimal time, minimal hazards, etc. Each of these parameters when minimized acts as the cost parameter. To each element of area $dl \times dl$ of the NS is associated the value dt of the cost of traveling the segment length dl on this area. An optimal path between source and destination is the one which yields $\min \int_{source}^{destination} dt$.

Navigation problems for one vehicle on a continuous surface as well as on a discrete grid have already been studied and solved^{1,2}. In our paper we consider navigation of more than one vehicle in a two dimensional space, where each vehicle has its own origin and destination. The objective is to navigate the vehicles in a way which minimizes the cost (time) of travel. The time of travel is the time passed between the earliest start time of one of the vehicles to the

last arrival at a destination. When two vehicles or more are involved in the problem, the requirement of collision avoidance may introduce a conflict between the optimal paths of the various vehicles. In order to resolve these conflicts the data base of possible paths is vastly extended and the search for optimal solution is very complicated. In a different study³ we have directly solved the one- and two-vehicle navigator in a multi speed discrete space. However we did not find a way to extend it as a practical technique for the general multi-vehicle navigator.

A simple NS terrain is defined with a binary speed. On this NS the speed of a vehicle, at each point, is either a positive constant or 0 (for an obstacle). The present study is an attempt to construct a multi-vehicle navigator, in binary speed space, using neural networks. By using neural networks one usually trades an optimal solution accomplished in "infinite" time with "good" solution accomplished in reasonable time.

This study is organized as follows: In section 2 we introduce the cost-surface space and the paths as graphs in this space. The cost-linking map is presented and we discuss the difference between paths solving a one-vehicle navigator and those solving multi-vehicle navigator. In section 3 the neural formulation is presented with the mapping of the space into neural variables, "neural paths" and equations. Section 4 contains the results of our simulations and section 5 the discussion.

2. Paths in the cost-surface space

A discrete representation of the surface of navigation is obtained by mapping the surface onto a graph as follows: a set of points $v(x, y)$ is chosen on the navigation space to be the nodes of the graph. Each node is connected by an edge to every other node which can be reached directly from it. To every edge is assigned a weight

which reflects the cost of traveling it. The edges can have two different weights for traveling it in opposite directions. A path is a sequence of adjacent directed edges from the origin to the destination.

Let us extend the NS terrain to a time-surface space, as shown in Figure 1. A path in this space is a sequence of edges, monotonic in t , between the source and destination. However, a legal path is one which obeys the restrictions set by the terrain. In order to get only legal paths, we construct a time-linking map³. This map assigns to every node the minimal time of travel needed to reach it from the origin. Using this map one can construct a graph of all the optimal paths from the source to all the nodes. This map specifies the t -coordinate of each node of the graph in the time-NS space. An optimal path for one vehicle, in the time-surface space, is single-valued in $v(x, y)$ and t . Namely, there is a one to one correspondence between $v(x, y)$ on the path and t . When more than one vehicle are involved each one of them has its own linking map. However, the optimal paths of two different vehicles may conflict. To avoid such a conflict one of the vehicles may be requested to postpone its arrival to or to detour the point of conflict. This imposition introduces paths which are not single valued in $v(x, y)$ and t as illustrated in Figure 2.

3. Neural formulation

Neural networks have been studied as an approach to various hard (NP-complete) optimization problems. Various applications have been investigated and explored^{4,5} since the work of Hopfield and Tank⁶. Here, we explore the possibility of using these massively parallel networks for the multi vehicle navigation problem.

The paths of the vehicles, as discussed above, are viewed as trajectories in the space-time. The space-time is mapped into neural variables in the following way: it is divided into a regular three dimensional lattice (x, y, t) . (For notational simplicity, we denote (x, y) by the vector x subsequently.) To each unit cell we associate a neural variable $\eta_i(x, t)$ whose desired

value, at stable state, is:

$$\eta_i(x, t) = \begin{cases} 1, & \text{if vehicle } i \text{ is at position } x \\ & \text{at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

A path is a sequence of neural variables with $\eta_i(x, t) = 1$ where t ranges from 0 to T and T is the time this path is traveled. A neural network is set up such that the neurons converge to a stable state which determines the paths as illustrated in Figure 3. A common practice in optimization by neural networks is to choose an energy function. However, finding the shortest paths is an iterative process, which makes our energy function time-dependent. Therefore, instead of minimizing an energy function we directly write down the equations relating the input "voltage" of the neurons to their output voltage. These equations impose the desired behavior of the neurons. Specifically, we have

$$du_i(x, t)/d\tau = C_1 + C_2 + C_3 + C_4 + C_5$$

where the first term evaluates the propagation of the path from the present position in the forward direction. The second term evaluates it with respect to the backward direction. The third term avoids head-on collision and the fourth term avoids swapping which occurs when two vehicles adjacent to each other switch positions. The fifth term forces one of the neighbors of an "on" neuron to be on, i.e. enforces continuity of the paths. In terms of neural variables, the dynamical equations is as follows:

$$\begin{aligned} du_i(x, t)/d\tau = & gate(i) \left(\right. \\ & - u_i(x, t) \\ & + A_1 \sum_{y \in Nb(x)} \eta_i(y, t-1) W_{xy} \\ & + A_2 \sum_{y \in Nb(x)} \eta_i(y, t+1) W_{y, x} past_i(x, t) \\ & \left. + A_3 \sum_k \eta_k(x, t) g(s_i(t) - s_k(t)) + \right) \end{aligned}$$

$$\begin{aligned}
& + A_4 \sum_k \sum_y \eta_k(x, t-1) \eta_i(y, t-1) \eta_k(y, t) \cdot \\
& \prod_{x' \in Nb(x), x' \neq y} \left(1 - \eta_i(x', t-1)(W_b - W_{xx'})/2\right) \cdot \\
& \prod_{x' \in Nb(y), x' \neq x} \left(1 - \eta_k(x', t-1)(W_b - W_{yx'})/2\right) \\
& + A_5 \sum_{y \in Nb(x)} f(\eta_i(y, t-1)) \cdot \\
& \prod_{x' \in Nb(y)} (1 - f1(\eta_i(x', t)))
\end{aligned}$$

Where every term $A_i\{\dots\}$ corresponds to the respective term C_i . $\eta_i(x, t) = h(u_i(x, t))$, $h(\cdot)$ is a sigmoid function giving the relation between the input and the output voltage of a neuron. $Nb(x) =$ neighborhood of x . W_{xy} is the cost of travel from x to y with regard to the destination of the vehicle. Namely, $W_{xy} = T(x) - T(y)$ where $T(u)$ is the time-linking map value of the node u . If $T(y) < T(x)$, it encourages the forward (in time) propagation of the path from x to y . $past_i(x, t) = \sum_{y \in Nb(x)} \eta_i(y, t-1)$ gates the

backward propagation. A neuron is affected by the future information only if it is a continuation of a path. $s_i(t) = \sum_x \eta_i(x, t)$, and $g(\cdot)$ is another sigmoid function which says that in case of collision, the vehicles with more possible paths should give way. The swapping term is most complicated. We leave out the detailed explanation except saying that $f(\cdot)$ and $f1(\cdot)$ are appropriately chosen highly nonlinear functions. Lastly, $gate(i) = \prod_{\tau \leq t} \eta_i(x_{di}, \tau)$ which stops the signal propagation for vehicle i once its destination x_{di} has been reached.

In the equations above, we encourage all possible paths to be stored in the states of the neurons. The redundancy in the formulation makes this possible. When the destinations are reached, we backtrack and choose one of the best paths computed by the network.

It is obvious that in the absence of collision, the paths obtained are the original optimal paths for a single vehicle where collisions are not considered.

Since the problem is inherently time dependent, the neuronal states at large t naturally wait for the information from neurons at smaller t . We may as well solve the equation for a fixed time window ω , namely we compute the paths

for the next ω moves. Then we repeat the procedure, calculating the paths piecewise until the destinations are reached.

4. Simulation results

We numerically integrated the above dynamical system, using a simple Euler method, in which case, synchronization does not have to be exactly enforced. Recall that an Euler solver for a differential equation $dx/dt = f(x)$ is an iterative mapping: $x_{i+1} = x_i + \epsilon f(x_i)$. This, together with the locality of the computational stencil enables us to parallelize the above algorithm very efficiently. If we go back to the equations above, the only global computation is computing $s_i(t)$, which can be obtained by locally updating the sum within each processor and combining the result in a binary tree. By iteratively solving a differential equation, exact synchronization is not needed because the dynamics is continuous. In a similar study¹¹, but slightly modified dynamic equations, almost a perfect speedup was obtained when it is implemented on the Meiko Computing Surface, a parallel machine with up to 32 transputer nodes as illustrated in Figure 4. The differential form also introduces some cooperation into the algorithm. This can be observed in the conflicting regions, like head-on collision and swapping, in which case the neurons iteratively adjust their values, trying to resolve the conflict.

5. Discussion

The neural net yields paths which slightly deviate from the optimal one-vehicle paths. This is because it is dominated by two main forces: one is the collision avoidance force and the other is the single vehicle optimal paths attractors. These attractors are the graphs determined by the linking map from a node to the destination. In our study of the two vehicle analytic navigator³ we are using an algorithm which updates this map. Applying this idea to the neural net system can improve the solutions obtained above. The neural net four vehicle navigator performs well, see Figure 3. However, with more vehicles and various possible paths for each it may perform less satisfactorily. Clearly we only presented a very initial study here. We need to look at much more complex problems including three dimensional navigation. We are

also looking into the elastic net ideas of Durbin and Willshaw⁷ as interpreted by Simic⁸ into neural networks. There are important analogies between track finding⁹, computer vision and navigation which we are exploring in an integrated research program¹⁰.

6. Acknowledgements

The work reported here is funded by the Department of Energy under grant DE-FG03-85ER25009, Program Manager of the Joint Tactical Fusion Program Office and National Science Foundation under grant EET-8700064.

7. References

1. M. Sharir & A. Schorr "On Shortest Paths in Polyhedral Spaces", SIAM J. Computing, 15,1, pp. 193-215, 1986.
2. J. S. B. Mitchell and C. H. Papadimitriou, "The Weighted Region Problem", Technical report, Dept. of Operations Research, Stanford University, 1986.
3. E. Gurewitz G. Fox, and Y. F. Wong "Parallel Algorithms for One and Two-vehicle Navigation" in this proceedings.
4. D. Z. Anderson, "Neural Information Processing Systems," American Institute of Physics, New York, 1988.
5. "IJCNN Proceedings," Washington D.C., 1989.
6. J. J. Hopfield & D. W. Tank, "Biol. Cybern. 52, pp. 141-152, 1985.
7. R. Durbin and D. Willshaw, Nature, 326, 689-691, 1987.
8. P. Simic, "Statistical Mechanics as the Underlying Theory of 'Elastic' and 'Neural Optimizers'," Technical Report C3P-787, contribution to 1989 SCS Eastern Conference, Florida, 1989.
9. Geoffrey Fox, "A Note on Neural Networks for Track finding," Technical Report C3P-748, 1989.
10. G. C. Fox, W. Furmanski, A. Ho, J. Koller, P. Simic, Y. Wong, "Neural Network and Dynamic Complex Systems," Technical Report C3P-695, 1989.
11. Y. F. Wong and G. C. Fox, "Use of Neural Network for Path Planning," Technical Report C3P-784, 1989.

* On a leave of absence from the Physics Department NRCN Beer-Sheva Israel.

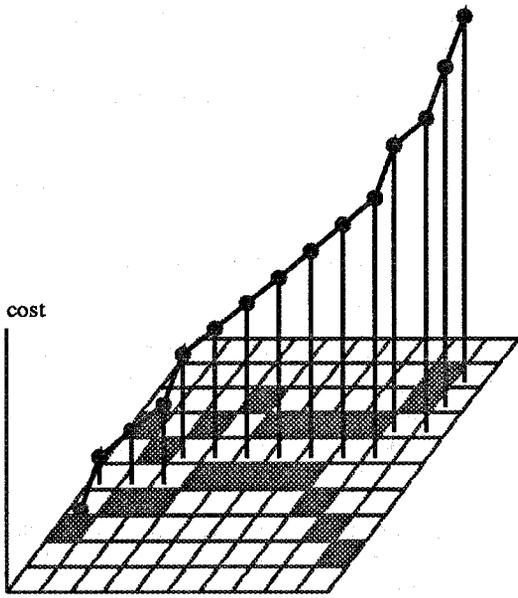


Figure 1. A path in the cost-terrain space

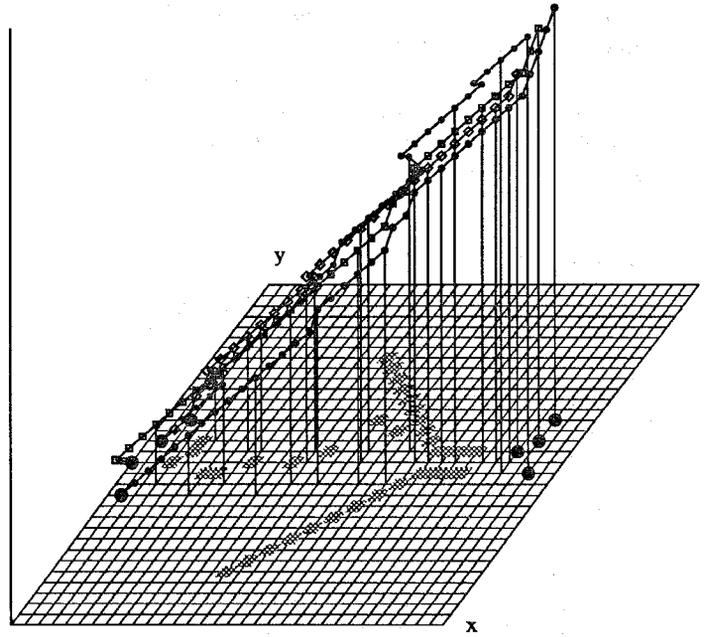


Figure 3. Four paths in the cost-terrain space calculated by the neural net

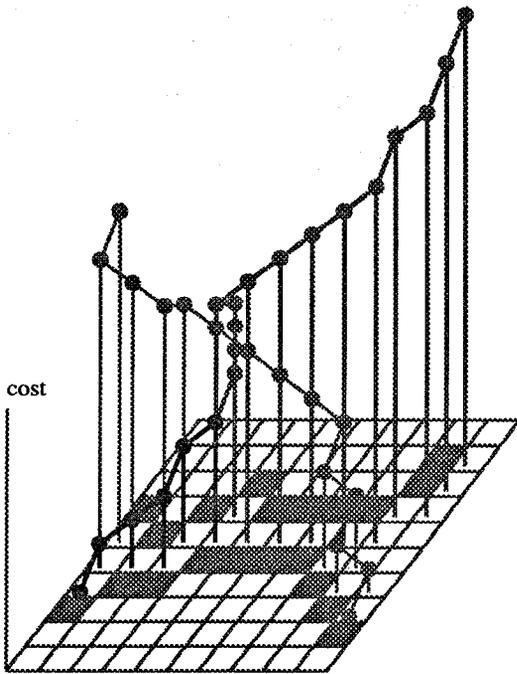


Figure 2. The two-vehicle navigator solution for a conflict imposing terrain and a path in the Cost-Terrain space.

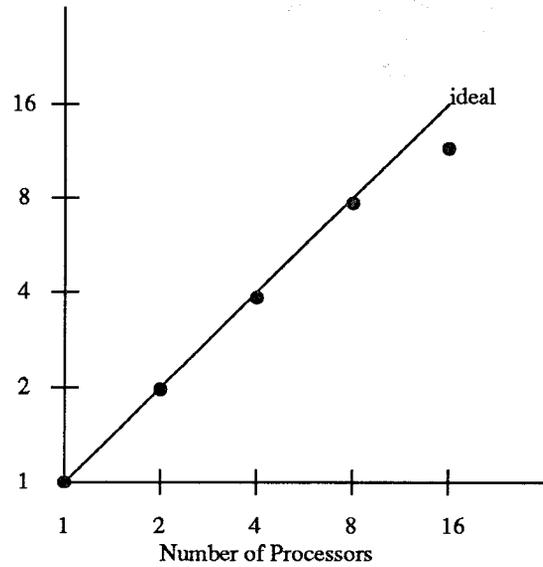


Figure 4. Speedup for 4 vehicle navigator running on 16-node Meiko Computing Surface